Saarbrücken Dissertations | Volume 40 |
in Language Science and Technology

Students' language in computer-assisted tutoring of mathematical proofs

Magdalena A. Wolska

Deutsches Forschungszentrum
für künstliche Intelligenz
German Research Center for
Artificial Intelligence

universaar
Universitätsverlag des Saarlandes
Saarland University Press
Presses Universitaires de la Sarre

Magdalena A. Wolska

Students' language in computer-assisted tutoring of mathematical proofs

Deutsches Forschungszentrum
für künstliche Intelligenz
German Research Center for
Artificial Intelligence

universaar
Universitätsverlag des Saarlandes
Saarland University Press
Presses Universitaires de la Sarre

Saarbrücken Dissertations
in Language Science and Technology
(Formerly: Saarbrücken Dissertations
in Computational Linguistics and Language Technology)

http://www.dfki.de/lt/diss.php

Volume 40

Saarland University
Department of Computational Linguistics and Phonetics

German Research Center for Artificial Intelligence
(Deutsches Forschungszentrum für künstliche Intelligenz, DFKI)
Language Technology Lab

D 291

Universitätsverlag des Saarlandes
Saarland University Press
Presses Universitaires de la Sarre

Postfach 151150; 66041 Saarbrücken
ISBN 978-3-86223-111-9 gedruckte Ausgabe
ISBN 978-3-86223-112-6 Online-Ausgabe
ISSN 2194-0398 gedruckte Ausgabe
ISSN 2198-5863 Online-Ausgabe
URN urn:nbn:de:bsz:291-universaar-1117

zugl. Dissertation zur Erlangung des Grades eines Doktors der
Philosophie der Philosophischen Fakultäten der Universität des Saarlandes
Berichterstatter: Prof. Dr. Manfred Pinkal, Prof. Dr. Michael Kohlhase
Kommissionsmitglieder: Prof. Dr. Dr. h.c. mult. Martin Kay, Dr. Stefan Thater
Dekan: Prof. Dr. Dr. h.c. Roland Marti
Tag der letzten Prüfungsleistung: 17.05.2013

Projektbetreuung universaar: Susanne Alt, Matthias Müller

Satz: Magdalena A. Wolska
Umschlaggestaltung: Julian Wichert

Gedruckt auf säurefreiem Papier von Monsenstein & Vannerdat

Bibliographische Information der Deutschen Nationalbibliothek:
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen
Nationalbibliografie; detailierte bibliografische Daten sind im Internet über
<http://dnb-d-nb.de> abrufbar.

Abstract

Truth and proof are central to mathematics. Proving (or disproving) seemingly simple statements often turns out to be one of the hardest mathematical tasks. Yet, doing proofs is rarely taught in the classroom. Studies on cognitive difficulties in learning to do proofs have shown that pupils and students not only often do not understand or cannot apply basic formal reasoning techniques and do not know how to use formal mathematical language, but, at a far more fundamental level, they also do not understand what it means to prove a statement or even do not see the purpose of proof at all. Since insight into the importance of proof and doing proofs as such cannot be learnt other than by practice, learning support through individualised tutoring is in demand.

This thesis has been part of an interdisciplinary project, set at the intersection of pedagogical science, artificial intelligence, and (computational) linguistics, which investigated issues involved in provisioning *automated* tutoring of mathematical proofs through dialogue in natural language (see Chapter 1). The ultimate goal in this context, addressing the above-mentioned need for learning support, is to build intelligent tutoring systems for mathematical proofs. The focus of this thesis is on the language that students use while interacting with such a system: its linguistic properties and computational modelling. Contribution is made at three levels: first, an analysis of language phenomena found in students' input to a (simulated) proof tutoring system is conducted and the variety of students' verbalisations is quantitatively assessed, second, a general computational processing strategy for informal mathematical language and methods of modelling prominent language phenomena are proposed, and third, prospects for natural language as an input modality for proof tutoring systems is evaluated based on collected corpora.

Proof tutoring corpora (Chapter 2)

In order to learn about the properties of students' language in naturalistic interactions with a tutoring system for proofs, two data collection experiments have been conducted. Both experiments were carried out in the so-called

Wizard-of-Oz (WOz) paradigm, that is, subjects interacted with a system simulated by a human. The interaction with the simulated system was typewritten. The language of the experiments was German; no constraints on the students' language production were imposed. Naïve set theory and binary relations were selected as the mathematical domains. In the set theory experiment, students were tutored using one of three tutoring strategies differing in the granularity of pedagogical feedback. In the binary relations experiment students were assigned to one of two experimental conditions: one group was shown study material formulated using mainly natural language (verbose), while the other group received mainly formalised content. The hypothesis was that the students' language would reflect the study material presentation format. The key lesson learnt from the experiments is that mathematics is a difficult domain for the Wizard-of-Oz setup. While WOz is an established research methodology in interactive systems, mathematics as a domain is challenging to the wizards due to the time-pressure on response generation related to maintaining a believable system setup. Certain interface features, in particular, the copy–paste mechanism and the ease with which it enables text reuse – in our case, stringing mathematical expressions together – produced substantial cognitive load on the wizards. In future experiments, support for the wizard, for instance, consisting of automated detection of mathematical expression errors, should be considered. The collected corpus comprises 59 dialogues with 1259 student turns and constitutes the source data for all the analyses.

Students' language in computer-based proof tutoring

Qualitative analysis (Chapter 3) The language of informal proofs in textbook discourse has been previously modelled based on mainly ad hoc analyses, rather than systematic corpus studies. The language of informal proofs has been described as precise, exhibiting no ambiguity and little linguistic variation, and consisting of stereotypical, formulaic phrasings in which natural language is used for the most part to express logical connectives. Contrary to these observations, our analysis of proof tutoring corpora shows that the language of students' proofs is rich in linguistic phenomena at all levels: lexical, syntactic, semantic, and discourse-pragmatic. The following utterances illustrate proof statements from our data:

$x \in B \implies x \notin A$

B enthaelt kein $x \in A$

B contains no $x \in A$

A hat keine Elemente mit B gemeinsam.
A has no elements in common with B.

A enthaelt keinesfalls Elemente, die auch in B sind.
A contains no elements that are also in B

$A \cap B$ ist $\in$ von $C \cup (A \cap B)$
$A \cap B$ is $\in$ of $C \cup (A \cap B)$

Nach der Definition von $\circ$ folgt dann (a, b) ist in $S^{-1} \circ R^{-1}$
By definition of $\circ$ it follows then that (a, b) is in $S^{-1} \circ R^{-1}$

wenn A vereinigt C ein Durchschnitt von B vereinigt C ist,
dann müssen alle A und B in C sein
If A union C is intersection of B union C, then all A and B must be in C

Students' input is for the most part highly informal and ranges from worded entirely in natural language, using a variety of syntactic constructions, through part-worded–part-formalised to entirely formalised; the longest mathematical expression consisted of 145 characters. Mathematical symbols and natural language are tightly interleaved and parts of mathematical expressions have to be interpreted in the context of natural language scope-bearing words (as in the second utterance). Symbols are also used as a kind of shorthand for natural language and wording can follow spoken language syntax when a formal expression is written down in its vocalised form (the last example). Moreover, natural language wording is imprecise, resulting in ambiguity in domain interpretation (e.g. ''contain'' as subset or membership). Discourse phenomena include domain-specific referring expressions (e.g. ''the left side'') and contextual operators (''analogously'', ''the other way round''). Since the use of mixed language and the imprecision phenomena are systematic, the key two requirements on a computational interpretation component are (i) integrating the semantic import of the symbolic expressions into the meaning of their cotext and (ii) representation of the imprecise concepts and an appropriate mapping to their mathematical interpretations. Frequently recurring complex clause structures in paratactic and hypotactic configurations call for a parsing method in which complex multi-clause utterances can be modelled with sufficient generality. For German specifically, the different word order in main clauses and subordinate clauses need to be modelled in a systematic way.

Quantitative analysis (Chapter 4) In order to assess the diversity in students' language production, a quantitative analysis of students' language has been carried out. First, a typology of students' utterances has been constructed. The typology focuses on solution-contributing utterances (utterances which directly or at a meta-level contribute to the proof being constructed), with the remaining subcategories grouped into one class (meta-level communication). Second, utterances have been preprocessed into verbalisation patterns which abstract away the specific mathematical expressions used and the domain terminology. Quantitative analysis is performed at three levels: the students contributions are characterised in terms of their language ‘‘modality’’ (natural language vs. symbolic notation), the binary relations corpus is characterised in terms of differences in the language production between the two study material conditions, and, finally, the distribution of utterance types in both corpora is analysed. Proof-contributing utterances are further analysed with respect to their function in the proof under construction (proof steps, declarations of proof strategy, etc.) and the type of content verbalised in natural language (logical connectives only, domain-specific vocabulary, etc.) Language diversity along these dimensions is quantified in terms of type–token ratios over the normalised linguistic patterns, frequency spectra, and pattern-vocabulary growth curves.

The conducted analyses show that the language of students’ discourse in proofs is not as repetitive as one might expect. Students use complex natural language utterances not only during meta-communication with the tutor, but also when contributing proof steps. The majority of utterances contain some natural language. Only 28 utterance verbalisations occurred in both data sets. Frequency spectra and the pattern growth curves show the degree to which the language is diverse. The majority of verbalisations are idiosyncratic (single-occurrence patterns). Not surprisingly, the majority of the meta-level communication are the students’ requests for assistance: requests for hints, definitions, explanations, etc. Interestingly, there is a relatively large number of discourse markers typical of spoken interaction. This suggests that participants had an informal approach to dialogue style and treated it much like a chat, adapting spoken language, which they would have otherwise used in a natural setting, to the experiments’ typewritten modality. The key conclusion from the analyses is that in a tutoring setting, even the seemingly linguistically predictable domain of mathematical proofs is characterised by a large variety of linguistic patterns of expression, by a large number of idiosyncratic verbalisations, and that the meta-communicative part of discourse which does not directly contribute to the solution has a conversational character, suggesting the students’ informal attitude towards the computer-based dialogues and their high expectations on the input interpretation resources. This calls for a combination of shallow and deep semantic processing methods for the discourse in question: shallow pattern-based approaches for contributions which do not add to the proof and semantic grammars for the proof-relevant content, in order to optimise coverage.

The analysis of the binary relations data revealed differences in the use of natural language and mathematical expressions between the two study material conditions. The verbose-material group tended to use more natural language than the formal-material group and the dialogue turns of the subjects in the verbose group contained more, but shorter, mathematical expressions. The formal material group tended to use longer formulas, and less natural language. Since the analysis of tutors' contributions showed no significant difference between the two conditions in the dialogue behaviour with respect to natural language and mathematical expression production, the differences in dialogue styles were at least partly due to the format of the study material presentation having a priming-like effect. These results have implications for the implementation of tutorial dialogue systems. On the one hand, more natural language, be it resulting from a verbose presentation of the study material or from the students' individual preference for a particular language style, imposes more challenges on the input understanding component. In the context of mathematics, this involves a reliable and robust parser and discourse analyser capable of interpreting mixed natural language and mathematical expressions. On the other hand, prompting for more symbolic language by presenting students with formalised material imposes stronger requirements on the mathematical expression parser since longer expressions tend to be prone to errors. The same holds of the copy–paste functionality: while convenient from the user's point of view, it may lead to mistakes of sloppiness while revising the copied text. This, in turn, calls for flexible formula parsing, error correction, and specific dialogue strategies to address formulas with errors.

Computational processing of informal proofs (Chapters 5 and 6)

Taking into account the range of linguistic phenomena in students' input and the need for a principled syntax–semantics interface for the proof contributing content, we propose a deep grammar-based approach to informal proof language. Processing of mixed language consisting of natural language words and mathematical expressions is achieved by abstracting over the symbolic notation in the course of parsing. Mathematical expressions are represented in terms of their syntactic types whose possible interactions with the natural language context is explicitly modelled in the grammar. Parsing is performed using a combinatory categorial grammar which builds a semantic dependency representation of the parsed input. The semantic representation is based on the Praguian notion of tectogrammatics, a language analysis level which considers the linguistic meaning of utterances, that is, meaning independent of their context. Tectogrammatical representations are further interpreted in the context of the mathematical domain in a stepwise fashion. First, imprecise lexemes are mapped to general concepts through a semantic lexicon. Then, the general concepts are mapped to mathematical domain concepts through a linguistically-motivated domain-ontology.

We propose methods of processing several language phenomena which systematically recur in the data, and which are critical for automated proof tutoring. This includes modelling basic syntactic phenomena (German word order in recurring constructions in mathematics, the mixed language, and the syntactic irregularities characteristic of the mathematical domain) and basic semantic imprecision phenomena. Moreover, we analyse a subset of interesting phenomena, which are not as frequent in the corpora, but which are highly complex from a computational processing point of view: the semantic reconstruction of the ''the other way round'' operator, reference to symbolic notation and propositions, and automated correction of mathematical expressions. Because the data is sparse, preliminary algorithms are proposed and evaluated in proof-of-concept studies or corpus studies are conducted as a preliminary step towards algorithm development. The processing methods proposed confirm that deep parsing using categorial grammars which build tectogrammatical (domain-independent) linguistic meaning representations of the analysed input, lends itself well to modelling a number of phenomena found in students' informal mathematical language.

Prospects for natural language-based proof tutoring (Chapter 7)

The final contribution of this work is a corpus-based performance assessment of the parsing component, the key part of the proposed input interpretation architecture. The collected corpora of learner proofs are used as data for an intrinsic evaluation which focuses on proof-contributing utterances. Grammars encoding verbalisation patterns are systematically tested in simulation experiments as follows: Grammars are built only based on utterances which *recur* in the development data. The recurring utterances stem from 42 dialogues. Parsers based on grammar resources constructed in this way are tested on an increasing number of dialogues. Performance is evaluated on two data sets: the data set constructed from utterances used for grammar development and on a blind set consisting of verbalisation patterns which occurred only once. Context-free grammars, developed and tested in the same manner, are used as baseline. Coverage (percentage of test set parsed) and parse ambiguity are reported.

The results show that hand-crafted semantic resources based on combinatory categorial grammars outperform context-free grammars on the coverage measures while remaining at a manageable ambiguity level. Moreover, they confirm our previous conclusion that the language used by students to talk about proofs is characterised by a large degree of diversity not only at a shallow level of specific phrasing, but also at a deeper level of syntactic structures used. Considering that only 59 dialogues have been available for analysis, we believe that the two corpora are insufficient, in the sense that they are not representative enough, for a robust proof tutoring system to be implemented at the present stage. First, the set of recurring verbalisations is small. This is against the intuition that the language of proofs should be small and repetitive. Grammars based on the set theory resources do not scale sufficiently even within-domain. Resources based on the binary relations data scale better within-domain, while, across-domains the difference in performance over within-domain data is negligible. More data would need to be collected in order to draw definitive conclusions. Interestingly, the results point at a methodological issue for WOz-based data collection strategy in the domain of proofs: Wizard-of-Oz experiments, logistically complex by themselves and in this case also cognitively demanding on the wizards, should cover multiple domains of mathematics rather than a single domain per experiment, as ours did, in order to provide more variety of proof verbalisations at one trial.

Nevertheless, considering that the promising coverage growth results are based on a small number of *partially* modelled dialogues, we also conclude that as far as language processing is concerned, natural language as the input mode for interactive proofs is a plausible alternative to menu-based input or structured editors, provided that more data and human resources for grammar development are available. We plan to conduct analogous linguistic analysis of authentic proofs appearing in mathematical publications in order to verify prior claims as to the linguistic proprieties of this genre and to apply processing methods proposed in this thesis in order to assess the prospects for automated knowledge extraction from scholarly mathematical discourse.

Zusammenfassung

Wahrheit und Beweis sind zentrale Teile der Mathematik. Die Wahrheit selbst scheinbar einfacher mathematischer Sätze zu beweisen (oder zu widerlegen) stellt sich oft als eine der schwierigsten mathematischen Aufgaben heraus. Dennoch wird in der Schule selten gelehrt, wie man Beweise führt. Studien zu kognitiven Schwierigkeiten beim Beweisen lernen, haben gezeigt, dass Studenten nicht nur formale Beweistechniken häufig nicht verstehen oder nicht anwenden können und nicht wissen, wie die formale mathematische Sprache zu benutzen ist, sondern sogar auf einer weitaus grundlegenden Ebene nicht verstehen, was es bedeutet, einen Satz zu beweisen, oder die Notwendigkeit, Beweise zu führen, überhaupt nicht einsehen. Da Einsicht in die Bedeutung des Beweises und Beweisen selbst nur durch Üben gelernt werden kann, ist Lernunterstützung durch individuelles Tutoring (Nachhilfe) gefragt.

Diese Arbeit ist Teil eines interdisziplinären Projektes, das an der Schnittstelle zwischen Pädagogik, künstlicher Intelligenz und (Computer-)Linguistik angesiedelt war und das sich mit der Untersuchung von *automatisiertem Tutoring* mathematischer Beweise in natürlichsprachlichem Dialog beschäftigt hat (siehe Kapitel 1). Das Fernziel in diesem Kontext, in Bezug auf den oben angesprochenen Bedarf nach Unterstützung beim Lernen, wäre die Entwicklung von intelligenten automatisierten Tutoring-Systemen für mathematische Beweise. Der Schwerpunkt dieser Arbeit liegt auf der Sprache, die die Studenten während der Interaktion mit einem solchen System verwenden: ihre sprachlichen Eigenschaften und ihre Modellierung mit dem Computer. Unser Beitrag findet auf drei Ebenen statt: Zuerst wird eine Analyse der sprachlichen Phänomene in den Studentenäußerungen zu einem (simulierten) tutoriellen System zum Beweisen durchgeführt und die Vielfalt der Verbalisierungen wird quantitativ bewertet. Als nächstes wird eine allgemeine Verarbeitungsstrategie für informelle mathematische Sprache und Methoden zur Modellierung von prominenten sprachlichen Phänomenen vorgeschlagen, und drittens werden die Perspektiven für natürliche Sprache als Eingabemodalität für ein tutorielles System für Beweise auf Grundlage von verfügbaren Korpora evaluiert.

Korpora zu mathematischem tutoriellen Dialog (Kapitel 2)

Um etwas über die Eigenschaften von Studentensprache in plausiblen Interaktionen mit einem tutoriellen System für Beweise zu lernen, wurden zwei Serien von Datenerhebungsexperimenten durchgeführt. Beide Versuche wurden im Rahmen des sogenannten Wizard-of-Oz (WOz)-Paradigmas durchgeführt, d.h. die Versuchspersonen interagieren mit einem System, das vollständig durch einen Menschen simuliert wird. Die Interaktion mit dem simulierten System geschah mittels Tastaturinput; es gab keine Einschränkungen bezüglich der Sprachproduktion der Studenten. Die Experimente fanden auf Deutsch statt. Als mathematische Domänen wurden naive Mengenlehre und binäre Relationen ausgewählt. Im Experiment zur Mengenlehre wurden Studenten mit je einer von drei tutoriellen Strategien unterrichtet. Diese unterscheiden sich in der Granularität des pädagogischen Feedbacks. Im Experiment zu binären Relationen wurden die Studenten einer von zwei experimentellen Bedingungen zugeteilt: eine Gruppe bekam Lehrmaterial gezeigt, das überwiegend in natürlicher Sprache (verbose) formuliert war. Die andere Gruppe erhielt hauptsächlich formalisierte Inhalte. Die Hypothese war, dass die Studentensprache die Präsentationsform des Lehrmaterials widerspiegeln würde. Die Haupterkenntnis aus den Experimenten ist, dass Mathematik für Wizard-of-Oz-Experimente eine schwierige Domäne ist. Obwohl WOz eine etablierte Forschungsmethode in der Entwicklung von interaktiven Systemen darstellt, ist die Aufgabe für den Wizard sehr anspruchsvoll. Dies ergibt sich aus dem Zeitdruck bei der Generierung von Systemantworten, der aus der Notwendigkeit resultiert, ein glaubwürdiges Setup aufrechtzuerhalten. Bestimmte Funktionalitäten der benutzten Schnittstelle, insbesondere der Copy-Paste-Mechanismus und die Leichtigkeit, mit der es die Wiederverwendung von Textbausteinen erlaubt – in unserem Fall mathematische Ausdrücke zusammenzustellen, – erzeugen eine zusätzliche kognitive Belastung des Wizards. In zukünftigen Experimenten sollte daher Unterstützung für den Wizard, zum Beispiel in Form von automatischer Erkennung von Fehlern in mathematischen Ausdrücken, berücksichtigt werden. Die gesammelten Korpora umfassen 59 Dialoge mit 1259 Studenten-Dialogbeiträgen.

Die Sprache der Studenten in computerbasierten Beweis-Tutoring

Qualitative Analyse (Kapitel 3) Die Sprache informeller Beweise wurde bisher nur in Lehrbuch-Diskursen untersucht und vor allem auf Grundlage von ad hoc Analysen modelliert. Sie wurde als präzise und stilistisch ''formelhaft'' beschrieben, zeige keine Mehrdeutigkeiten und wenig sprachliche Variation und bestehe aus stereotypischen Formulierungen, in denen natürliche Sprache hauptsächlich dazu benutzt werde, logische Verknüpfungen auszudrücken. Im Gegensatz zu diesen Beobachtungen zeigt unsere Korpusanalyse, dass die Sprache der Studentenbeweise reich an sprachlichen Phänomenen auf allen Ebenen ist: lexikalisch, syntaktisch, semantisch und diskurs-pragmatisch. Die folgenden Äußerungen zeigen beispielhaft Aussagen aus Beweisen in unseren Korpora:

$x \in B \implies x \notin A$

B enthaelt kein $x \in A$

A hat keine Elemente mit B gemeinsam.

A enthaelt keinesfalls Elemente, die auch in B sind.

$A \cap B$ ist $\in$ von $C \cup (A \cap B)$

Nach der Definition von $\circ$ folgt dann (a, b) ist in $S^{-1} \circ R^{-1}$

wenn A vereinigt C ein Durchschnitt von B vereinigt C ist,
dann müssen alle A und B in C sein

Die Äußerungen der Studenten sind überwiegend informell und reichen von rein in natürlicher Sprache mit einer Vielzahl von syntaktischen Konstruktionen, über teils-in-Worten-teils- formal-formuliert bis hin zu vollständig formalisiert; der längste mathematischen Ausdruck bestand aus 145 Zeichen. Mathematische Symbole und natürliche Sprache sind eng miteinander verflochten und Teile von mathematischen Ausdrücke müssen im Kontext skopustragender natürlichsprachlicher Wörter interpretiert werden (die zweite Äußerung). Symbole werden auch als eine Art Kurzschrift für natürliche Sprache verwendet und der Wortlaut folgt mitunter der Syntax gesprochener Sprache, wenn ein formaler Ausdruck in der Form geschrieben wird, wie er auch gesprochen wird (das letzte Beispiel). Darüber hinaus ist der Wortlaut natürlicher Sprache ungenau, was zu Unklarheiten bei der Interpretation innerhalb der Domäne führt (‘‘enthalten’’ als Teilmenge oder Element einer Menge). Diskursphänomene beinhalten domänenspezifische referierende Ausdrücke (z.B. ‘‘die rechte Seite’’) und kontextuelle Operatoren (‘‘analog’’, ‘‘ umgekehrt’’). Da die Verwendung von gemischter Sprache und die Ungenauigkeitsphänomene systematisch sind, sind die zwei wichtigsten Anforderungen an eine Komponente zur automatischen Interpretation (i) die Integration des semantischen Gehalts der symbolischen Ausdrücke in die Bedeutung ihres Kontextes und (ii) die Repräsentation der ungenauen Konzepte und eine entsprechende Zuordnung zu ihrer mathematischen Interpretationen. Häufig wiederkehrende komplexe Satzstrukturen in parataktischer und hypotaktischer Konfigurationen erfordern eine Analysemethode, bei der komplexe Äußerungen aus mehreren Teilsätzen in ausreichend allgemeiner Form modelliert werden können. Für das Deutsche im Speziellen müssen die verschiedenen Wortstellungen in Haupt- und Nebensätzen in systematischer Weise modelliert werden.

Quantitative Analyse (Kapitel 4) Um die Vielfalt bei der Sprachproduktion der Studenten zu beurteilen, wurde sie quantitativ analysiert. Zunächst wurde eine Typologie der Studentenäußerungen konstruiert. Die Typologie konzentriert sich auf die zur Lösung beitragenden Äußerungen (Äußerungen, die zu dem aktuellen Beweis direkt oder auf einer Meta-Ebene beitragen), während die restlichen Unterkategorien alle zu einer Klasse (Meta-Ebene-Kommunikation) zusammengefasst werden. Als nächstes wurden Äußerungen zu Verbalisierungsmustern vorverarbeitet, die von den spezifischen mathematische Ausdrücken und der spezifischen Terminologie der Domäne abstrahieren. Eine quantitative Analyse wird auf drei Ebenen durchgeführt: Zunächst wird die Studentensprache in Bezug auf die sprachliche ''Modalität'' (natürliche Sprache vs. symbolische Notation) charakterisiert. Das Korpus zum Thema binäre Relationen wird in Bezug auf Unterschiede in der Sprachproduktion zwischen den beiden Lehrmaterialstypen charakterisiert. Schließlich wird die Verteilung der Äußerungsarten in beiden Korpora analysiert. Zum Beweis beitragende Äußerungen werden darüber hinaus mit Bezug auf ihre Funktion im aktuellen Beweis (Beweisschritte, Erklärungen der Beweisstrategie, usw.) und die Art der Inhalte, die in natürlicher Sprache verbalisiert sind (nur logische Verknüpfungen, domänenspezifisches Vokabular, usw.), analysiert. Die Sprachvielfalt entlang dieser Dimensionen wird durch das Type-Token-Verhältnis über den normalisierten sprachlichen Muster, Frequenzspektren und Wachstumskurven von Mustervokabular quantifiziert.

Die Ergebnisse zeigen, dass die Sprache im Studentendiskurs über Beweise nicht so repetitiv ist, wie man erwarten könnte. Studenten verwenden komplexe natürlichsprachliche Äußerungen nicht nur während der Meta-Kommunikation mit dem Tutor, sondern auch, wenn sie Beweisschritte beitragen. Die Mehrzahl der Äußerungen enthält zumindest teilweise natürliche Sprache. Nur 28 Verbalisierungen von Äußerungen traten in beiden Datensätzen auf. Die Frequenzspektren und die Muster-Wachstumskurven zeigen das Ausmaß der Vielfalt in der Sprache. Die Mehrheit der Verbalisierungen sind individuell und treten nur ein einziges Mal auf. Es ist nicht überraschend, dass die Mehrheit der Studentenäußerungen auf Meta-Ebene Bitten um Hilfe sind: um Hinweise, um Definitionen, um Erläuterungen usw. Interessanterweise gibt es eine relativ große Anzahl von Diskursmarker, die typisch für gesprochene Interaktion sind. Dies deutet darauf hin, dass die Teilnehmer eine informelle Einstellung gegenüber dem Dialogstil hatten und ihn ähnlich wie einen Chat behandelt haben, indem sie gesprochene Sprache für den geschriebenen Dialog adaptiert hatten, die sie sonst in einer Situation mit einem menschlichen Tutor verwendet hätten. Die wichtigste Schlussfolgerung aus den Analysen ist, dass in einem tutoriellen Kontext auch die scheinbar sprachlich vorhersehbare Domäne mathematischer Beweise durch eine große Vielfalt sprachlicher Ausdrucksmuster und eine große Anzahl von idiosynkratischen Verbalisierungen geprägt ist, und dass der meta-kommunikative Anteil des Diskurses, der nicht direkt zur Lösung beiträgt, Konversationscharakter hat, was die informelle Haltung der Studenten gegenüber dem computerbasierten Dialog und ihre hohen Erwartungen an den Ressourcen zur Eingabeinterpretation nahelegt. Dies erfordert eine Kombination von flachen und tiefen semantischen Verarbeitungsmethoden für den Diskurs: flache musterbasierte Ansätze für diejenigen Beiträge, die nicht zum Beweis führen, und semantische Grammatiken für die beweisrelevanten Inhalte, um die Abdeckung zu optimieren.

Die Analyse der Daten zu binären Relationen ergab Unterschiede in der Nutzung von natürlicher Sprache und mathematischen Ausdrücken zwischen den beiden Lehrmaterialstypen. Die Gruppe, die wortreiches Lehrmaterial bekam, verwendete tendenziell mehr natürlichsprachliche Ausdrücke als die Gruppe, die formelreiches Lehrmaterial bekam. Auch enthält das sprachliche Material der Probanden der Gruppe mit wortreichem Lehrmaterial mehr, aber kürzere mathematische Formeln. Die Gruppe mit formelreichem Lehrmaterial dagegen benutzte tendenziell längere Formeln, dafür aber weniger natürliche Sprache. Da die statistische Analyse der Tutorenbeteiligung keinen signifikanten Unterschied im Dialogverhalten des Tutors in Bezug auf die Produktion natürlichsprachlicher versus mathematischer Ausdrücke zwischen den beiden Versuchsgruppen zeigte, sind diese Unterschiede im Dialogstil zumindest teilweise auf die Form der Lehrmaterialspräsentation zurückfürbar; der Lehrmaterialtyp scheint eine Priming-Wirkung auf die Sprachproduktion der Probanden gehabt zu haben. Die Testergebnisse über den Einfluss der Lehrmaterialspräsentation haben Auswirkungen auf die Implementierung von tutoriellen Dialogsystemen. Auf der einen Seite stellt der intensive Gebrauch von natürlicher Sprache, sei es aufgrund einer wortreichen Präsentation des Lehrmaterials oder individueller Präferenzen des Studenten für einen bestimmten Sprachstil, eine Herausforderung für das Eingabeanalysemodul eines Dialogsystems dar.

Fürs Verstehen der Fachsprache der Mathematik wird ein zuverlässiger, robuster Parser sowie ein Diskursanalysemodul benötigt, das in der Lage ist, eine Mischung aus natürlichsprachlichen und mathematischen Ausdrücken zu interpretieren. Wenn man, auf der anderen Seite, die Studenten dazu anregt, eine formelreiche Sprache zu benutzen, indem man ihnen entsprechendes Lehrmaterial zeigt, wachsen dadurch die Anforderungen an den Parser für mathematische Ausdrücke, weil längere Ausdrücke tendenziell fehleranfälliger sind. Das gleiche gilt für die Copy-Paste-Funktionalität: Auch wenn diese Eingabehilfe aus der Sicht des Benutzers praktisch ist, kann sie zu Flüchtigkeitsfehler bei der Überarbeitung von kopiertem Text führen. Dies wiederum erfordert eine flexible Syntaxanalyse mathematischer Formeln, Fehlerkorrektur und spezifische Dialogstrategien für den Umgang mit fehlerbehafteten Formeln.

Computerbasierte Verarbeitung informeller Beweise (Kapitel 5 und Kapitel 6)

Unter Berücksichtigung der Bandbreite linguistischer Phänomene in der Eingabe seitens der Studenten und der Notwendigkeit einer prinzipiellen Syntax-Semantik-Schnittstelle für Inhalte, die zum Beweis beitragen, schlagen wir einen Ansatz zur Verarbeitung informeller Beweissprache vor, der auf dem Formalismus der Tiefengrammatik beruht.

Die Analyse der natürlichen Sprache gemischt mit mathematischen Ausdrücken wird durch Abstraktion von Formeln im Verlauf des Parsings erreicht. Mathematische Ausdrücke werden durch ihre möglichen syntaktischen Typen repräsentiert, deren Wechselwirkungen mit dem natürlichsprachlichen Kontext explizit in der Grammatik modelliert werden. Der Parsingvorgang wird unter Verwendung einer kombinatorischen Kategorialgrammatik ausgeführt, die eine semantische Dependenzrepräsentation der analysierten Eingabe erstellt. Die auf dieser Weise erhaltene semantische Struktur gründet auf Tektogrammatik, eine von der Prager Schule postulierte multistratale Sprachanalyse, die sprachliche Bedeutung von Äußerungen unabhänging von ihren Kontext betrachtet. Tektogrammatische Darstellungen werden dann schrittweise in Bezug auf ihre mathematische Domäne interpretiert. Zunächst werden ungenaue Lexeme mit Hilfe eines semantischen Lexikons auf allgemeine Konzepte abgebildet. Dann werden allgemeine Konzepte durch eine sprachlich motivierte Ontologie auf Konzepte der mathematischen Domäne abgebildet.

Es werden Sprachverarbeitungsmethoden vorgeschlagen für Phänomene, die systematisch in den Daten wiederholt auftreten und somit entscheidend für ein automatisiertes Unterrichten von mathematischen Beweisen sind. Dazu gehört die Modellierung grundlegender syntaktischer Phänomene (Wortstellung in wiederkehrenden Konstruktionen in der Mathematik, gemischte Sprache, und syntaktische Unregelmäßigkeiten als Merkmal der betrachteten Domäne) und grundlegende Phänomene von semantischer Ungenauigkeit. Darüber hinaus wird eine Teilmenge von interessanten Phänomenen analysiert, die zwar nicht zahlreich in Korpora aufzufinden, jedoch aus Sicht der

Computerverarbeitung sehr komplex sind: die semantische Rekonstruktion des ‘‘umgekehrt’’-Operators, das Verweisen auf symbolische Notation und Propositionen, sowie das Korrigieren mathematischer Ausdrücke. Da die Daten spärlich sind, werden vorläufige Algorithmen vorgeschlagen und in Proof-of-Concept-Studien evaluiert. In einigen Fällen werden Korpusstudien als erster Schritt zur Entwicklung von Algorithmen durchgeführt. Die Verarbeitungsmethoden bestätigen, dass tiefensyntaktische Analyse mit Kategorialgrammatiken, die domänen-unabhängige Repräsentationen sprachlicher Bedeutung der analysierten Eingabe aufbauen, sich gut zur Modellierung einer Reihe von Phänomenen in der informellen mathematischen Sprache der Studenten eignen.

Perspektiven natürlichsprachlicher Beweis-Tutor-Systeme (Kapitel 7)

Der letzte Beitrag der vorliegenden Arbeit ist eine korpusbasierte Leistungsbewertung der Parser-Komponente, also des wesentlichen Bestandteils der vorgeschlagenen Strategie zur Eingabe-Analyse. Die gesammelten Korpora von Lernerbeweisen werden als Datensammlung für eine intrinsische Auswertung herangezogen, die auf solche Äußerungen im Dialog abzielt, die zum Beweis wesentlich beitragen. Grammatiken, die Versprachlichungsmuster kodieren, werden systematisch in Simulationsexperimenten wie folgt getestet: Grammatiken werden nur auf Grundlage von Äußerungsmustern erstellt, die in den ausgewählten Arbeitsdaten wiederholt vorkommen. (Die wiederkehrenden Äußerungen stammten aus 42 Dialogen.) Parser, die auf so gebauten Grammatikressourcen basieren, wurden auf einer zunehmenden Zahl von Dialogen getestet. Die Leistung wurde auf zwei Datensätzen ausgewertet: ein Datensatz, der aus Äußerungen gebaut wurde, die für die Grammatik-Entwicklung genutzt wurde, und ein Blind-Satz bestehend aus Verbalisierungsmustern, die nur einmal aufgetreten sind. Kontextfreie Grammatiken, die in der gleichen Weise entwickelt und getestet wurden, wurden als Baseline verwendet. Abdeckung (Anteil des Test-Sets, das geparst werden kann) und Parser-Mehrdeutigkeit werden angegeben.

Die Ergebnisse zeigen, dass manuell erstellte semantische Ressourcen auf der Basis kombinatorischer Kategorialgrammatiken kontextfreien Grammatiken überlegen sind, was die Abdeckung angeht, aber dennoch ein noch handhabbares Maß an Ambiguität aufweisen. Außerdem bestätigen sie unsere bisherige Schlussfolgerung, dass die Sprache, die Studenten verwenden, um über Beweise zu sprechen, von einem großen Maß an Vielfalt gekennzeichnet ist, nicht nur auf einer flachen Ebene von spezifischen Formulierungen, sondern auch auf der tieferen Ebene der benutzten syntaktischen Strukturen.

Da nur 59 Dialoge für die vorliegende Untersuchung zur Verfügung standen, glauben wir, dass die beiden Korpora unzureichend sind, in dem Sinne, dass sie zum aktuellen Zeitpunkt nicht repräsentativ genug sind für die robuste Implementierung eines Dialogsystems fürs Lehren mathematischer Beweise. Erstens ist die Menge von wiederkehrenden Sprachmustern klein. Dies widerspricht der Intuition, dass die Sprache der Beweise klein und repetitiv sein sollte. Grammatiken, die auf Ressourcen zur Mengenlehre basieren, lassen sich selbst innerhalb der gleichen Domäne nicht gut übertragen. Ressourcen auf Grundlage der Daten von binären Relationen sind besser innerhalb der Domäne übertragbar, doch der Unterschied zur Performanz in fremden Domänen ist vernachlässigbar. Mehr Daten müssten gesammelt werden, um endgültige Schlüsse zu ziehen. Interessanterweise deuten die Ergebnisse auf eine methodische Frage für WOz-basierte Datenerfassungsstrategien im Bereich von Beweisen hin: Wizard-of-Oz Experimente, die per se logistisch komplex und in diesem Fall auch kognitiv anspruchsvoll für den Wizard sind, sollten mehrere Domänen innerhalb der Mathematik abdecken, nicht nur eine einzige Domäne pro Experiment, wie im der vorliegende Studie. Dadurch würde man eine größere Vielfalt von Beweisverbalisierungen erzielen. Wenn man aber bedenkt, dass die vielversprechenden Ergebnisse zur Abdeckung einer immer wachsenden Anzahl von linguistischen Phänomenen auf einer relativ kleinen Anzahl von *teilweise* modellierten Dialoge fußen, stellen wir dennoch fest, dass, was die Sprachverarbeitung angeht, die natürliche Sprache als Eingabe-Modus für interaktive Beweise eine plausible Alternative zu menübasierter Eingabe oder Struktur-Editoren ist, vorausgesetzt, dass sowohl mehr Daten als auch mehr Fachläute für Grammatikentwicklung zur Verfügung stehen. Wir planen, unter anderem, analoge linguistische Analysen von authentischen Beweisen durchzuführen, die in mathematischen Publikationen erschienen sind, um Behauptungen bezüglich linguistischer Eigenschaften dieses Genres zu prüfen und um die Perspektiven für einen automatisierten mathematischen Wissenserwerb aus dieser Art von Diskurs zu beurteilen.

Acknowledgements

I had the enormous luck of working in the group of Prof. Manfred Pinkal. I'd like to thank Manfred for this opportunity and for his support throughout the years. The atmosphere in Saarbrücken was truly stimulating. I think it's thanks to how Manfred always encourages people to work *together*, at the same time supports them in pursuing their *own* interests and projects (admittedly, this here should have been finished much earlier, but I learnt so much along the way!), and at the same time still, takes care that people feel engaged in forming a little departmental *community*. I'd also like to thank Manfred for being my *Doktorvater*, for his always constructive critique, feedback, and thought-provoking research discussions.

I would like to equally thank Prof. Michael Kohlhase: for reviewing my thesis, for his helpful comments and suggestions, and for the inspiring conversations at the CICMs. His enthusiasm for research is infectious and motivational!

Thank you to the members of my thesis committee, Prof. Martin Kay, Prof. Roland Marti, and Dr Stefan Thater, for an interesting discussion at the viva. Stefan I would also like to thank for being such a great company as an office neighbour over the many years.

Andrea Horbach and Ciprian Gerstenberger I'd like to very much thank for the help with the German abstract.

Many thanks to Frau Alt and Frau Kopp from universaar and SULB – as well as everyone else involved in the publication process – for the friendly and efficient support in bringing this volume into existence.

To all my friends and colleagues at the Computational Linguistics and Phonetics and DFKI a thank you for the company and for what I learnt from you research-wise and other-wise. Special thanks to Werner Saurer for the fun of co-teaching *Semantics of tense and aspect* and *Mathematical logic*. Another special thank you to Ciprian again for the time we worked and hung out together. (I still giggle when I reread what's commented out in our paper and our grammars!)

It’s been a *decade*, but I still do want to mention that I truly enjoyed working and interacting with as well as learning from the people in the then Prof. Jörg Siekmann’s group: Jörg himself, Chris Benzmüller, Chad Brown, Helmut Horacek, Henri Lesourd, Martin Pollet, Marvin Schiller, and Bao Vo, who were also involved in the DIALOG project. It was a great research environment, interesting work, and fun at the same time! Special thanks to Helmut, who was my main collaborator at the time, for being such a kind person, so easy to work with, and willing to share his vast knowledge. A lot of what’s in this thesis is based on our collaboration. To Chad notable thanks for our seminal joint work on *plannotation* (‘‘still’’ unpublished!) and on formalising the Declaration of Human Rights. And to Henri for the looong conversations about learning, principles, and the academia, the kind which you never know on what topic they end up and how they got there, that continued into and throughout the Allegro project.

Very special thanks to Sowmya Vajjala and Martí Quixal for all the discussions about research and the universe. In verse! (And prose.) Tübingen would have been a misery without you!

Michałowi: za wikt i opierunek. And for putting up with me in general...

No amount of paper would suffice to write down thanks to my Parents, Jadwidze i Włodzimierzowi Wolskim – all the good in me stems from them, all the bad is ‘‘accomplishment’’ of my own – but I know I can bring little smiles to their faces with these two words: *Waść skończył!*

How happy is the little stone
That rambles in the road alone,
And doesn’t care about careers,
And exigencies never fears;
Whose coat of elemental brown
A passing universe put on;
And independent as the sun,
Associates or glows alone,
Fulfilling absolute decree
In casual simplicity.

Emily Dickinson

Contents

List of Figures

List of Tables

Introduction

Why can't Johnny prove?

Dreyfus suggests that there are possibly two main reasons: proving is unlike any calculation-oriented task that students are confronted with before they get to the point where proofs become the central mathematical activity. The transition to the kind of knowledge needed for proving is complex and difficult; especially since the criteria for judging acceptability of proofs are not clear cut (Dreyfus, 1999).[1] Multiple other educational studies which attempted to understand the cognitive mechanisms involved in learning to do proofs and the obstacles that learners encounter in the process, showed that fundamental difficulties arise for students already in recognising the very nature of proof, that is, what a proof is and its role in mathematics (Bell, 1976; Michener, 1978; Chazan, 1993; Moore, 1994; Sierpinska, 1994; Anderson, 1996; Almeida, 2000; Hanna, 2000, among others). This is not surprising, since, from a pedagogical point of view, there is little agreement on the notion of proof even among mathematicians and mathematics teachers (Davis and Hersh, 1981; Hersh, 1997a; Knuth, 2002) and the role of proof and the criteria of proof's validity vary between mathematics foundations (Hanna, 1995). There is also little agreement as to the pedagogical methods suitable for teaching to do proofs. Almeida (2000) points out that while for mathematicians a proof is the culminating point in theory development which involves *intuition*, *trial*, *error*, *speculation*, *conjecture*, and finally *proof*, in university courses students encounter a rather different model: *definition*, *theorem*, *proof*. As a result, students tend to think of proofs merely as exercises in demonstration and explanation rather than as a way of gaining insight into a problem. They exhibit ''a lack of concern for meaning, a lack of appreciation of proof as a functional tool'' (Alibert and Thomas, 1991), sometimes even do not recognise the need for proof at all (Dreyfus, 1999; Almeida, 2000; Selden and Selden, 2003), or merely recognise

[1] Do check out the reference for the source of the opening line.

that they are supposed to give ‘‘some’’ proof (Almeida, 2000). Hersh summarises this situation as follows:

> When you’re a student, professors and books claim to prove things. But they don’t say what’s meant by ‘‘prove.’’ You have to catch on. Watch what the professor does, then do the same thing.
>
> Then you become a professor, and pass on the same ‘‘know-how’’ without ‘‘knowing what’’ that your professor taught you. (Hersh, 1997b, p. 50)

The symbolic notation is only a low-level factor which, however, often also constitutes a serious cognitive barrier in understanding mathematical concepts (Moore, 1994; Dorier et al., 2000; Booker, 2002; Downs and Mamona-Downs, 2005).

Whatever Johnny’s fundamental problem in grasping the point of proof, an uncontroversial claim is this: *all* mathematics teachers would agree that key in acquiring proving skills is practice. Practise, practise, practise! One just has to do a lot of proofs. Well, what if Johnny could practise doing proofs on his computer?...

The project of which this work was part aimed at realising this very idea. It investigated the *issues involved in provisioning intelligent computational systems which would help students learn to do proofs the way that a good teacher would do it: by engaging a student in an argumentative dialogue, trying to guide him towards discovering a reasoning path leading to a proof*. Tutoring interactions of this kind, involving flexible dialogue and encouraging self-explanation, have been shown to improve learning (Moore, 1993), whereas natural language would mitigate problems with mathematical notation identified by Moore (1994) by letting students ‘‘capitalise’’ on their skills and ‘‘compensate’’ for weaknesses: students unskilled in notation could still get credit for valid proofs. In this work we address one aspect of the project: *the language of informal mathematical discourse, its linguistic properties and computational processing*. We situate the problem in the context of three scenarios in which understanding the language of proofs is relevant: tutoring, interaction with automated mathematics assistance systems, and document processing. We focus, however, on students’ language in the context of tutoring.

Generally speaking, the term ‘‘mathematical discourse’’ may be broadly understood to refer to any kind of discourse which concerns mathematics: from scientific discussions among mathematicians or classroom discussions between students and teachers, through mathematical textbooks and scientific publications, to popular science prose. The discourse may be concerned with analysis of historical developments in mathematics, the evolution of understanding of mathematical concepts and of the language used to name them, discussions of examples, explications of mental representations (ways of thinking about a concept), or simply *statements of mathematical facts*. Steenrod, Halmos, Schiffer, and Dieudonné (1973) and Bagchi and Wells (1998) refer to the latter kind of mathematical discourse as the *mathematical register*.

Bagchi and Wells loosely define mathematical register as ‘‘text in a natural language, possibly containing embedded symbolic expressions, [that] communicates mathematical reasoning and facts directly’’. Since mathematical register focuses on mathematical facts and the formal structure, it is presumed that ‘‘statements in mathematical register [can] be translated into a sequence of statements in a formal logical system such as first order logic’’ (*ibid.*) Examples of mathematical register include mathematical definitions, statements of theorems, and proofs of theorems. The core contributions of this thesis concern mathematical discourse – mathematical register – in this narrow sense.[2]

The most prominent surface characteristic of mathematical discourse is its familiar mixture of symbols and natural language. While, in principle, proofs can be presented using the symbolic mathematical language alone – as in formal logic, for instance – this presentation style is not common in communicating mathematics. Halmos (1970) argues that symbolic notation does not have to dominate in a proof for it to make a ‘‘better’’ proof.

Support for open-ended natural language in proof tutoring systems requires that the language understanding component be capable of building a symbolic representation of the learners’ input which can be translated into an input representation of a deduction system responsible for reasoning tasks. With the view to provisioning such input processing capabilities we collected a corpus of learner proofs constructed in natural language interactions (in German) with an anticipated dialogue-based tutoring system simulated by a human. Using qualitative and quantitative analysis methods, in this thesis we attempt to answer the following questions based on this data:

- What language phenomena emerge in naturalistic dialogues with a proof tutoring system?
- Does the range of linguistic verbalisations tend to be limited or is the language diverse? Is the students’ language affected by the way the study material is presented?
- Given the range of language phenomena in informal mathematical discourse, what is an appropriate approach to processing this kind

[2] Whenever we use the term ‘‘mathematical discourse’’ or ‘‘mathematical language’’, we have in mind mathematical register as defined here and its language, respectively. Other types of mathematical discourse are outside the scope of this work.

of language? What semantic representation provides the appropriate meaning abstraction for further semantic processing of the identified language phenomena?

- Can a systematic procedure be defined which would take informal proof steps as input and return as output a representation suitable for translation to a domain reasoner's language? What parameters are involved? What processing subcomponents and resources are needed?
- What is the prospect for automated tutoring of proofs in natural language?

We show that students' language in computer-assisted tutoring of mathematical proofs is rich in complex linguistic phenomena (Chapter 3) and characterised by a large variety of verbalisations, and that students tend to use the kind of language that they are confronted with in the learning materials they use for study (Chapter 4). Based on the insights from the linguistic analysis, we propose an architecture for computational processing of proof language based on a deep semantic grammar and a strategy for processing the mixed natural and symbolic language typical of mathematics (Chapter 5). We show how to model selected recurring phenomena systematically in a semantic framework and propose initial algorithms for those complex phenomena which would require further data collection for a more thorough analysis and evaluation (Chapter 6). Finally, we show that the grammar formalism on which our language processing architecture crucially relies, provides good generalisations in modelling linguistic phenomena (Chapter 7), which lets us conclude that the language modelling strategy we propose is a viable contribution towards computational processing of informal mathematical discourse.

Parts of the work presented here had been published in collaborative articles. Material from the following previously published articles is included:

Chapter 2: (Benzmüller et al., 2003a; Wolska et al., 2004b; Benzmüller et al., 2006)
Chapter 3: (Benzmüller et al., 2003b)
Chapter 4: (Wolska and Kruijff-Korbayová, 2006a; Wolska, 2012)
Chapter 5: (Wolska and Kruijff-Korbayová, 2004a,b; Wolska, 2008; Wolska et al., 2010)
Chapter 6: (Wolska and Kruijff-Korbayová, 2004a; Wolska et al., 2004a; Gerstenberger and Wolska, 2005; Horacek and Wolska, 2005a,c; Wolska and Kruijff-Korbayová, 2006b; Horacek and Wolska, 2006a,b,c)

Chapter 1

Background and related work

This chapter introduces the project within which this work has been set and summarises the state of the art in modelling mathematical discourse. We start by presenting target scenarios envisaged for our approach to computational interpretation of mathematical language. After introducing the basic notions relevant when talking about discourse processing in our domain, a high-level architecture of a system for processing mathematical language in the target scenarios is outlined. The tasks of each of the system components are briefly summarised. The reminder of the chapter is dedicated to a discussion of related work. We briefly report on processing user input in an existing proof tutoring system, on formal models of mathematical language, implemented systems for processing mathematical discourse, controlled natural languages for mathematics, and annotation of mathematical proofs. The chapter closes with a discussion of implications for our approach.

1.1 Target scenarios

The research reported in this thesis stems from a larger project, DIALOG, whose objective was an empirical investigation into the issues involved in modelling natural language interaction with a mathematics assistance system (Pinkal et al., 2001, 2004).[1] While the core focus of the DIALOG project was on interactive tutoring, the linguistic analysis of mathematical language, the language interpretation methods we propose and the evaluation results we report are relevant in the context of processing mathematical discourse in general, be it tutorial dialogue or mathematical prose. We envisage three application scenarios and larger architectures in which they can be applied.

[1] DIALOG was part of the ‘‘Resource-adaptive cognitive processes’’ Collaborative Research Centre funded by the Deutsche Forschungsgemeinschaft as Sonderforschungsbereich 378.

The first scenario and the main motivation of this work, formulated in the introduction, is *computer-based interactive tutoring of mathematical proofs* and is related to the project from which this work stems. The ultimate goal in this context is the provision of systems for tutoring mathematical proofs by means of flexible dialogue in natural language. Target users of such systems are learners of mathematics and mathematics teachers contributing proof exercises. The linguistic material which needs to be interpreted in this context are the utterances which learners enter while communicating with the system, be it proof steps or meta-level speech acts, such as requests for explanation of domain concepts. The second, related scenario is *interactive proof construction with the help of human-oriented automated deduction systems*. The goal in this case is the provision of natural language user interfaces for theorem provers, possibly embedded within larger mathematical document authoring environments. Potential users of such applications are mathematicians or teachers preparing course materials or textbooks. Different variants of this scenario might involve not only different degrees of linguistic richness, but also different degrees of interaction flexibility: the proof language might be unconstrained or it might be a controlled natural language, proofs might be constructed either incrementally step by step, each step being verified at a time (much like in interactive proof assistance systems) or complete proofs could be checked at once as self-contained discourses. The linguistic material to be interpreted in this context are proof steps of different complexity constructed by a user of an automated deduction system, a mathematician or a student. The third scenario involves *computational processing of mathematical documents*, textbooks or scientific publications, such as those found in arXiv,[2] the online preprints archive. The goal in this case is to enable search, information retrieval, and knowledge extraction in scholarly mathematical documents. Computational interpretation of proof discourse in this context would be a step towards transforming documents into a machine-understandable representation and, in a further perspective, towards automated verification of published proofs. While no interactive proof construction is involved here, this scenario involves authentic mathematical discourse as it is routinely written and published by mathematicians. In terms of authenticity of the linguistic material it is therefore closer to the first scenario and rather more challenging than the second.

Common to the three scenarios is that, ultimately, mathematical content expressed in natural language – mathematical proofs – needs to be processed by a reasoning component, an automated theorem prover or a proof checker, in order to verify its validity. Previous work in the latter scenario relied on a dedicated reasoning system whose proof representation language directly

[2] `http://www.arxiv.org`

reflected the discourse structure representation used for modelling proofs at the linguistic level (Zinn, 2006). By contrast to this work, we do not assume that the reasoner is directly linked with the language understanding component by means of its internal representation. Instead, we construct a symbolic representation of the linguistic meaning of a proof discourse fragment and rely on a dedicated procedure to interface between this representation and a formal proof representation required by one of the *existing* automated deduction systems. Below we outline a general architecture of a system for interactive processing of natural language proofs in the scenarios mentioned above.

1.2 High-level system architecture

Before presenting the architecture, we introduce the terminology which we will use when talking about the system's components and the interpretation strategy.

The macrostructural discourse unit of interest in our scenarios is a proof. In the context of a mathematical document, it could be of course another mathematical discourse entity, such as a definition of a concept, a statement of a theorem. An elementary discourse unit in a proof is a proof step which can consist of a number of elements (an assertion, inference rule(s), etc.) The basic notions relevant in the context of discourse/dialogue processing are a communicative unit, a contribution, and a discourse model:

Communicative unit

By a communicative unit we mean a scenario-specific unit of communication from the point of view of the macrostructure of the discourse under analysis. In the dialogue-based tutoring scenario a communicative unit is a dialogue turn which a learner composes while interacting with the tutoring system. In the interactive proof construction scenario, depending on the mode of user interaction, a communicative unit may be a single sentence which constitutes a proof statement or a multi-sentence discourse segment which constitutes an entire proof. In the document processing scenario, it is a discourse segment which comprises an entire proof in a document. As an *elementary communicative unit* we consider a clause. A communicative unit may consist of one or more utterances in a dialogic discourse or sentences in a narrative discourse.

Contribution, Proof contribution

In dialogue and conversation analysis *contribution* is a basic unit, a segment ‘‘contributed’’ by a dialogue participant. It is often used synonymously with the term ‘‘turn’’. A turn may consist of one or more *utterances*, intentional, meaningful communicative acts in an interaction. In the tutoring scenario, utterances adding information to the solution being constructed we will call *solution-contributing utterances*. A *proof contribution* is a solution-contributing utterance which expresses proof-relevant content, that is, one or more proof steps or parts thereof. A more detailed typology of utterances in the (proof) tutoring scenario will be presented in Chapter 4. More generally, contributions which express domain-relevant content we will call *domain contributions*. Examples of domain contributions include solution-contributing utterances or students’ requests for explanation of a concept, for instance: ‘‘What is a powerset?’’.

Discourse model A discourse model is a symbolic representation of the discourse structure built up incrementally out of (parts of) discourse segments. It represents discourse segments’ semantics and discourse-level relations (for instance, rhetorical relations) between segments or parts thereof. By semantics of a discourse segment we mean its linguistic (domain-independent) *and* domain-specific interpretation. In particular, it is possible that the former is known (has been constructed), while the latter is not (domain interpretation of the linguistic content could not be assigned). Depending on the linguistic content of the discourse segments, discourse relations between segments or elementary units may be unknown (underspecified) as well. In dialogic discourse, a discourse model is part of a dialogue model, which is, in turn, a symbolic representation of the dialogue structure and includes a model of the state of the dialogue at any point of interaction and a model of the dialogue flow.

Independently of the scenario, we assume that mathematical language interpretation is part of a larger modular mathematical discourse processing architecture whose components perform specialised tasks specific to the scenarios outlined above. Figure 1.1 depicts the place of the language interpretation process within a system for processing mathematical discourse, be it dialogic or narrative. The language interpretation process operates on communicative units. In this thesis, we focus on the semantic processing of a subset of *proof contributing utterances*. The process comprises a number of subprocesses whose purpose is to build a symbolic representation of proof contributions’

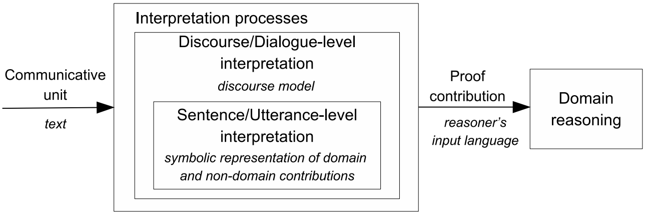

Communicative unit
text
Interpretation processes
Discourse/Dialogue-level interpretation
discourse model
Sentence/Utterance-level interpretation
symbolic representation of domain and non-domain contributions
Proof contribution
reasoner's input language
Domain reasoning

Figure 1.1: Language interpretation process in the overall architecture

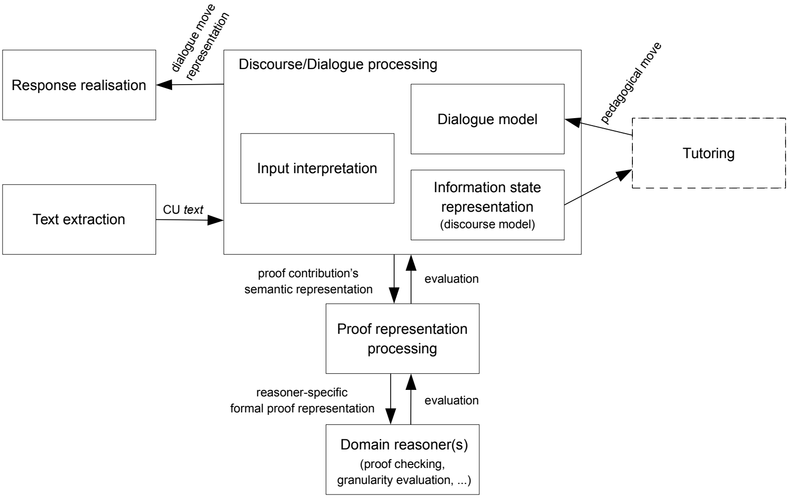

dialogue move representation
Response realisation
Discourse/Dialogue processing
Dialogue model
Input interpretation
Information state representation (discourse model)
pedagogical move
Tutoring
Text extraction
CU text
proof contribution's semantic representation
evaluation
Proof representation processing
reasoner-specific formal proof representation
evaluation
Domain reasoner(s) (proof checking, granularity evaluation, ...)

Figure 1.2: High-level architecture of a system for processing mathematical discourse

semantics both at the domain-independent and the domain-specific levels. In the approach we propose, these representations mediate between the textual natural language presentation of the proof contributions and a formal proof representation language constructed at the interface to a domain reasoner. Figure 1.2 schematically presents a generalised view of an architecture of a computational system for processing mathematical discourse in the context of the described scenarios. It comprises the core modules of such a system, including components specific to the different scenarios. Modules common to the three scenarios are marked with solid lines. The module marked with dashed-lines, Tutoring, is an additional module specific to the tutoring scenario. The language interpretation processes are part of the input interpretation module; ‘‘input’’ is a communicative unit relevant in the given scenario. Semantically processed contributions are incorporated into a discourse model and, subsequently, the relevant domain-level units (proof steps, parts thereof, or entire proofs) are translated into a formal language of a reasoner. Below we elaborate on the tasks of each of the architecture components.

Text extraction The purpose of the text extraction module is to identify and isolate the linguistic material relevant for analysis. Text extraction operates at the interface between the input acquisition module (a graphical user interface (GUI), for instance) and the input interpretation module. Its task is to deliver the text of communicative units in a format which the language interpretation module expects. This may involve stripping unnecessary markup from the original input or extracting the relevant units from a larger mathematical discourse (for instance, extracting proofs from a mathematical document).[3]

Input interpretation In general, the task of the input analysis module is two-fold. First, it is to construct a representation of the linguistic meaning and the domain interpretation of the input contributions. Second, given the linguistic meaning and depending on whether the contribution has an interpretation in the mathematical domain (is a domain contribution), it is to identify and separate within the contribution's symbolic representation the parts which convey proof steps (proof contributions), and thus should be passed on to a reasoner, and the parts which a reasoner does not process, but which, in case of the tutoring scenario, should be processed directly by the dialogue processing component. The core focus of this thesis is an interpretation strategy for proof contributions and will be discussed in more detail in Chapters 5, 6 and 7.

Discourse/dialogue processing Discourse processing addresses pragmatic (in the technical sense of the word) phenomena, that is, semantic phenomena beyond the level of compositional semantics of an utterance and the lexical meaning of words from which the utterance is composed. This includes processing discourse cohesion phenomena (resolving referring expressions, etc.), rhetorical phenomena (identifying rhetorical relations between elementary discourse units), discourse structure phenomena (identifying larger segments expressing a certain purpose), and recognising the illocutionary force of utterances (the functional role of utterances in a discourse).

[3] We include this process for the sake of completeness, however, we do not address it any further in this work. Likewise, we do not address user interface issues. We assume that the input to the language processing component contains only proof contributions, that is, one or more utterances which convey proof steps or parts thereof.

In a dialogue processing architecture a discourse model is a part of a *dialogue model*, a structured representation of the state of dialogue at any point of interaction, the so-called ‘‘information state’’, and of the flow of interaction in the given domain. The latter is a representation of dialogue structure which controls the dialogue progression and specifies ways in which the information state is to be updated following each contribution. Dialogues may be represented as frame structures (see, for instance, (Bobrow et al., 1977)), state transition graphs (e.g., (Metzing, 1980; McTear, 1998)), information state representations with update rules (Traum et al., 1999), a combination of those (e.g., a state transition graph with information state update rules (Lemon and Liu, 2006; Horacek and Wolska, 2005d; Buckley and Wolska, 2007, 2008a)), or as a probabilistic system (e.g. (Young, 2000)). The purpose of the dialogue structure model is to drive the interaction forward by selecting a dialogue move to be contributed following a contribution of a dialogue system’s user.[4]

Proof representation processing Proofs are structured discourses whose core elements are mathematical statements along with references to other statements which justify the validity of the inferences; these may be theorems or lemmata, or previously inferred statements. Proofs may be expressed in an informal language admitting of arbitrary natural language verbalisation, as in textbooks or mathematical publications, or in a formally defined language, as in formal logic or automated deduction systems. Linguistic properties of informal proof language will be discussed in Chapter 3. Proof discourse understanding consists in, firstly, understanding the language of the discourse and, secondly, recognising, understanding, and verifying the validity of (i) the individual statements, (ii) the relations between them, and (iii) the macrostructure of the proof. The latter involves, for instance, identifying the justifications of proof

[4] A dialogue move is a dialogue contribution which expresses a communicative intention, for instance, that of requesting information or requesting that some action be performed (a command). Examples of taxonomies of dialogue moves developed for dialogue and dialogue systems research include DAMSL (Allen and Core, 1997), DATE (Walker and Passonneau, 2001), or DIT++ (Bunt, 2009). The notion of a dialogue move stems from the notion of a speech act (Austin, 1962; Searle, 1999). In speech act terms, ‘‘information request’’ or ‘‘command’’ describe the utterance’s *illocutionary force*, that is, the speaker’s intention expressed in uttering certain words. Some dialogue contributions have an implicit or explicit meta-level communicative function of facilitating the maintenance of the state of knowledge shared between dialogue participants, the ‘‘common ground’’. These contributions are called ‘‘grounding moves’’ and include, for instance, requests for clarification or acknowledgements. *Grounding* is a meta-communicative process in conversational interaction which interlocutors employ to establish whether the other party has understood what has been said as intended (Isaacs and Clark, 1987; Clark and Schaefer, 1989; Clark and Brennan, 1991; Clark, 1996). See, for instance, (Traum, 1994; Matheson et al., 2000; Li et al., 2006; Bunt et al., 2007) for research on computational models of grounding.

steps (be it those explicitly stated or those left implicit) and the larger reasoning structure into which statements are organised; this structure may result from the choice of the proof method, as in, for instance, proofs by induction or proofs by cases. Proof representation processing is concerned with both of these aspects: the proof language and the proof structure.

The first proof representation related task is to mediate between the symbolic representation of the proof contributions constructed by the language understanding module and that of a domain reasoner. Introducing a dedicated interface between these representations ensures modularity of the overall architecture and a clear separation of linguistic processing and domain reasoning (see Section 5.1 for further motivation of the interpretation architecture design). From a practical point of view, this task consists in defining a translation between the symbolic representations of proof contributions produced by the language understanding process and the language of an automated prover or proof checker which serves as the domain reasoner.

The second proof representation processing task is to build and maintain a representation of the proof constructed in the course of dialogue: of the statements themselves, the relations between them, and of the overall structure of the proof. This may, moreover, involve storing *correctness* evaluations of proof contributions, obtained from a domain reasoner, or other evaluations relevant in deciding on further actions; for instance, *granularity* or *relevance* evaluations. In the tutoring scenario, proof contributions evaluated as invalid or inappropriate in the given context may also need to be stored in order to provide the tutoring module with information which may be useful in deciding on the immediate response and an overall pedagogical strategy to adopt.[5]

Domain reasoning By domain reasoning in the context of the presented scenarios we mean theorem proving. Generally speaking then, a domain reasoner needed for this task is an automated deduction system, however, the detailed task specification is dependent on the scenario and its requirements.

Automated deduction has been an active research area of artificial intelligence for over 30 years. Many automated theorem provers exist, however, not all of them can be immediately used in the scenarios in question. Theorem provers differ in their proof automation capabilities (the extent to which they can make inferences or produce entire proofs automatically), in the requirements as to the level of detail in the proofs they can verify (whether they can

[5] In the DIALOG project publications we referred to the module performing proof representation processing tasks in the software systems' jargon as the ''Proof Manager''. More details on the proof structure processing tasks can be found in (Benzmüller and Vo, 2005; Benzmüller et al., 2009) and on the issues related to automated evaluation of granularity in (Schiller et al., 2008).

reason at the level of abstraction at which humans do, in particular, whether they can infer omitted proof steps and parts of proof steps; this is related to the previous point: automation capabilities), and in the type of information they can provide about the automatically inferred steps (for instance, whether they can be queried about inference rules applied in an automated derivation). They also differ in the formal languages in which proofs must be specified in order to be processed. In fact, there is no ''standard'' proof language for deduction systems. This is due, among others, to the fact that the systems are based on different underlying mathematical foundations – set theory, type theory – which ''speak'' different languages. The high-level representations proposed by Autexier et al. (2004) and Autexier and Fiedler (2006) are ''assertion-level'' representations which admit of underspecification typical of proofs produced by humans. The differences in the input languages to theorem provers is the main reason why dedicated translations into specific proof languages are needed; in our architecture, this translation is the responsibility of the proof representation processing module discussed above.

Without making claims as to which existing theorem prover would be best suited for the considered scenarios, the requirements on the reasoner can be summarised as follows: In the document processing scenario, a proof checker would be needed for a proof verification task. The proof checker would have to handle human-oriented underspecified proofs. The interactive theorem proving scenario would require a proof checker, although a fully-fledged theorem prover would certainly be of help to a proof author.[6] Tutoring is perhaps the most demanding of the three scenarios because of the properties of proofs produced by learners. First, similarly to the other scenarios, learner proofs tend to omit proof steps or parts of proof steps, therefore mechanisms of reconstructing missing proof parts are necessary. Second, learners are prone to producing false proof steps, therefore, fast falsification is required. Third, special functionality may be needed in order to support tutoring, for instance, in deciding on whether a contributed correct step is relevant in the given proof context and of appropriate granularity, or in generating tutoring hints. In the DIALOG project Ωmega (Siekmann et al., 2003) was intended as the reasoning system. More details on this system and on how it was adapted to support the kinds of proofs which students produced in our experiments and tutoring itself can be found in the following publications: (Vo et al., 2003; Autexier et al.,

[6] We are not aware of large scale evaluations of theorem provers handling formalisations of proofs published in mathematical articles nor of theorem provers supporting natural language input; however, see (Wagner et al., 2007) for an attempt in this direction and (Vershinin and Paskevich, 2000; Verchinine et al., 2008) and Naproche (`http://www.naproche.net` [Accessed: 2006]) for controlled natural language approaches.

2004; Benzmüller and Vo, 2005; Autexier et al., 2009; Benzmüller et al., 2009; Autexier et al., 2012). Proofs from the corpora collected in the project were also used in a case study with Scunak (Brown, 2006a,b).

Tutoring In the tutoring scenario, the tutoring module is responsible for the pedagogical aspects of the interaction. It is the tutoring module that decides which dialogue move is to be performed once a learner's contribution has been grounded[4] at the communicative level and how it should be realised by the generation process (discussed below).

Automated tutoring relies on symbolic or probabilistic models of pedagogical strategy which, effectively, drives the dialogue. In order to decide on a dialogue action, a pedagogical strategy model typically refers to the history of the learner's performance in prior and current interactions or assessments, the so-called *learner/student model* (VanLehn, 1987; Elsom-Cook, 1993). The teaching strategy itself may comprise a static model of pedagogical knowledge on tutoring in the given domain (see, for instance, (Rosé et al., 2001; Zinn et al., 2003; Fiedler and Tsovaltzi, 2003; Tsovaltzi et al., 2004)) or a complex adaptive symbolic or stochastic model which adjusts its behaviour based on, among others, interaction variables and a learner model (Dzikovska et al., 2007; Forbes-Riley and Litman, 2009; Tsovaltzi, 2010). Recent work on pedagogical strategy models for intelligent tutoring systems takes into account such aspects of interaction as learner's uncertainty as well as affect and emotions in tutoring (see, for instance, (Litman and Forbes-Riley, 2004; D'Mello et al., 2007; Porayska-Pomsta et al., 2008)).

Response generation/Realisation The complexity of the response generation task, the categories of responses and their form, depends on the scenario. In the case of the tutoring domain it also depends on the adopted pedagogical strategy, since it directly influences the range of dialogue moves needed to realise the pedagogical dialogue actions; which may, in turn, also influence the range of dialogue moves contributed by learners during interaction. Response types may range from simple acknowledgements, through evaluative or corrective feedback, to hints of various complexity; for instance, hints on omitted proof elements in the document processing scenario or pedagogical hints motivated by a teaching strategy in the tutoring scenario. Dialogue move taxonomies for the tutoring scenario have been proposed, for instance, in (Marineau et al., 2000; Porayska-Pomsta et al., 2000; Tsovaltzi and Karagjosova, 2004; Wolska and Buckley, 2008; Campbell et al., 2009).

The standard language generation process comprises three phases, each of which involves a number of substeps (Reiter and Dale, 2000): (i) *content* *and structure determination*, that is, selection of information, communicative goal(s), to be communicated and selection of the larger structure in which it should be communicated, (ii) sentence/utterance planning or so-called *microplanning*, that is, lexical selection, syntactic structure selection, etc., and (iii) *surface realisation*, that is, producing the surface form of the utterance(s) to be communicated from the representation constructed in the previous two steps (putting the abstract representations of communicative goals into words). In the tutoring context, the first two phases are of course influenced by the tutoring process and the pedagogical strategy it realises. In particular, a pedagogical strategy might not only determine the pedagogical content and the dialogue moves to be communicated at a given dialogue state, but also influence the high-level decisions as to how a pedagogically motivated communicative goal is to be broken down into atomic communicative goals, how these atomic goals should be related to one another in rhetorical terms, down to specifying the lexemes to be used as well as the mood and mode of the utterance(s) to be generated. We do not elaborate any further here on the generation process itself nor on the methods employed in building language generation components in tutorial dialogue systems because in the overall architecture the generation process does not interact directly with the semantic interpretation process which is the main focus of our work. However, further discussion of response generation issues in the context of mathematics tutoring can be found, for instance, in (Callaway et al., 2006), while issues involved in natural language verbalisation of proofs, for instance, in (Huang and Fiedler, 1997; Holland-Minkley et al., 1999; Horacek, 2001a; Fiedler, 2005).

The processes outlined above constitute the core of an architecture for mathematical discourse processing for the scenarios we introduced in the beginning of this chapter. A complete computational system would of course include processes and components which we will not discuss here. Their purpose and functionality would depend on the larger application scenario. For instance, in the tutoring scenario the proof tutoring system might be embedded in a larger environment for learning mathematics, such as LeActiveMath (Melis et al., 2001, 2006) which is itself a complex system incorporating dedicated components for curriculum development, exercise sequencing, learner modelling, and others. In the interactive proof construction scenario, proofs might be constructed in a mathematical document authoring environment with sophisticated mathematical expression editing capabilities, requiring a complex graphical interface; as in, for instance, (Wagner et al., 2007; Wagner and Lesourd, 2008). Finally, mathematical document processing for knowledge extraction, information retrieval, and semantic search, would necessitate a range of components providing support for content-oriented services, such as management of digital libraries of mathematical documents and storage of mathematical knowledge in structured repositories, both of which are active areas of research in the Mathematical Knowledge Management and Digital Mathematical Libraries communities. The arXMLiv project, aiming at migrating arXiv documents into an XML-based representation, is an example of an effort in this direction.[7]

While the described scenarios are diverse in terms of their purposes, the functionalities they are intended to offer, the users they target, and, possibly, the language style of their proof discourses (more or less verbose), they all require that the mathematical language is computationally processed in order to enable automated proof checking. In the following section we give a brief overview of related work on modelling and processing mathematical language.

1.3 Related work

The early history of attempts at building systems for natural language mathematics – Abrahams' Proofchecker (1963), Bobrow's STUDENT (1964), and Simon's Nthchecker (1990) – has been summarised in (Zinn, 2004). We do not repeat the summaries here and refer to Zinn's dissertation's Section 2.1 for an overview. In this section, we briefly outline related work on modelling mathematical discourse by pointing out five directions of this research: (i) interactive natural language tutoring of proofs, exemplified by the EXCHECK system, (ii) formal (theoretical) models of mathematical discourse, (iii) implemented systems for processing mathematical discourse, (iv) controlled natural languages for proofs, and (v) proof annotation languages.

1.3.1 Natural language proof tutoring with EXCHECK

Patrick Suppes' group at the Stanford University Institute for Mathematical Studies in the Social Sciences (IMSSS) were among the pioneers in *large-scale* computer-assisted instruction (CAI). The IMSSS research on computer-based teaching of mathematics dates back to the 1960s[8] and has encompassed a multitude of domains, including, aside from various areas of mathematics, Slavonic languages, music, and computer programming. In fact, the IMSSS systems from the 1970s and their successors have continued being used in university-level tutoring; for instance, the VALID system for symbolic logic (Suppes,

[7] XML, eXtensible Markup Language, is a generic document encoding scheme for machine-readable documents (`http://www.w3.org/TR/REX-xml` [Accessed: 2006]). Further information on arXMLiv can be found at `http://kwarc.info/projects/arXMLiv` [Accessed: 2006].

[8] The early history of this research is reported in (Suppes, 1974).

1981) and its successors at the Carnegie Mellon University (Scheines and Sieg, 1994) or the EPGY proof environment at Stanford (McMath et al., 2001).

EXCHECK is one of the IMSSS systems developed in the mid-70s. Since then, different versions of the system have taught Stanford students in university-level courses on elementary logic, axiomatic set theory, and proof theory.[9] Much like in our experiments, a student working with EXCHECK would be presented with lesson material in the domain of interest (set theory, for instance) and asked to prove theorems from this domain.

EXCHECK was designed with specific goals in mind (see (Smith and Blaine, 1976)) two of which are most relevant here. First, it was intended to serve as a base and a practical laboratory for research on natural language processing. Mathematics was chosen as the domain of foremost interest because on the one hand, its semantics is well-understood, while on the other hand, informal mathematics and its language offer interesting research problems from the point of view of both automated problem solving as well as natural language processing. Second, proof tutoring was intended to be realised at a level appropriate for human problem solving, rather than driven by the requirements of an underlying proof checking system. Already at the time of EXCHECK did the IMSSS researchers observe that informal proofs, in particular, students' proofs, substantially differ from formal proofs which can be verified by proof checkers or constructed by automated deduction systems. EXCHECK was intended to bridge this gap and as such was among the first, if not *the* first automated system addressing human-level theorem proving.[10] The DIALOG project was in fact driven by the same motivations and goals as those behind the EXCHECK research (Benzmüller et al., 2009).

There is a number of interesting aspects to the EXCHECK system and similarities with the tutoring system for mathematical proofs envisioned in the DIALOG project.[11] First, EXCHECK allows students to construct proofs in an interactive manner. That is, the system and a student engage in a dialogue in which the student constructs a proof with the help of the system step by step. Second, EXCHECK proofs can be formulated in a ‘‘natural style’’ which

[9] Numerous articles related to the IMSSS research on CAI, in particular the EXCHECK system, are available on Suppes' corpus website (`http://suppes-corpus.stanford.edu` [Accessed: 2006]). It would be impractical to cite all the relevant published work here because the resulting list of references would probably be almost as long as this chapter itself. Therefore, we only cite those papers which specifically address or mention those aspects of the systems which are of particular interest here; that is, language and dialogue processing. An overview of the systems and of empirical studies during the first decade of the systems' use can be found in (Suppes, 1981).

[10] See (Autexier et al., 2004; Benzmüller and Vo, 2005; Autexier and Fiedler, 2006) for a discussion on human-oriented proofs and automated deduction in the context of the DIALOG project.

[11] As a convention we will use present tense when talking about the EXCHECK from the 70s; even if the modern EXCHECK-based systems differ from the original version in functionality.

Table 1.1: Fragment of the EXCHECK input language; examples of formal expressions (left) and their natural language verbalisations (right)

Formal	Informal
`(A x)(E y)(x sub y)`	For every x there is a y such that x is a subset of y
`Function(F) & F:A -> B`	F is a function and F maps A into B
`{x :  x neq x} = 0`	The set of all x such that x is not equal x is empty

`(A A){Dinfinite(A) IFF`
`(E C)(C psub A & C := A)}`
	For all A, A is Dedekind-infinite just in case there is a C such that C is a proper subset of A and C is equipollent to A

`(∀x)(x ∈ A → x ∈ B)`	For all x, if x is in A then x is in B
`(∀x)(x ∈ B)`	For all x, x is in B

is close to the standard mathematical practice. In particular, proofs can be informal in the sense that not all the steps of reasoning must be specified. Moreover, EXCHECK admits of certain flexibility in the language style of the input: proofs can be written using either symbolic mathematical expressions or in ‘‘mathematical English’’. Table 1.1 shows examples of inputs which it can interpret: both symbolic expressions as well as their corresponding natural language verbalisations are shown; see (Smith and Blaine, 1976; McDonald, 1981) (punctuation and capitalisation preserved).

As the examples illustrate, the range of complexity of the EXCHECK input statements is quite broad, encompassing from simple to compound formulas as well as utterances formulated entirely in natural language. This coverage is achieved by explicit authoring of input utterances, that is, specifying the language fragment by means of a grammar. EXCHECK has been conceptualised as an environment for both authoring proof exercises and tutoring. As part of the exercise authoring process a content developer must define a language fragment to talk about the mathematical theory in question, that is, formulate natural language sentences, such as those exemplified in Table 1.1, which a student can use. This is done by explicitly writing a context free grammar for the anticipated language fragment as well as ‘‘macro templates’’ which transform the parse outputs directly into the internal representation of the proof checker. The language processing component of EXCHECK, CONSTRUCT, is presented in detail in (Smith, 1974) and (Smith and Rawson, 1976). While there are limitations on the complexity of the natural language which can be interpreted by the system (for instance, the utterance ‘‘Everything is in B’’, which is a possible paraphrase of the licensed input utterances ‘‘$(\forall x)(x \in B)$’’ and ‘‘For all x, x is in B’’, cannot be parsed), the EXCHECK/CONSTRUCT system is the most impressive of the implemented systems, considering its coverage and the fact that the system has been *actually used* in teaching proofs; see (Suppes and Sheehan, 1981) and the other reports on the university-level computer-assisted instruction at Stanford on the Suppes’ corpus website.

1.3.2 Formal analysis of mathematical language

By formal analysis of mathematical language we mean systematic studies on formal semantics of the mathematical register. Within this line of research, Fox (1999) focuses on certain ‘‘non-schematic’’ occurrences of variable letters in mathematics which cannot be modelled in the standard way as referring expressions and proposes to extend theories of discourse interpretation for mathematics with Fine’s theory of arbitrary objects (Fine, 1983).

Ganesalingam (2009) gives a formal analysis of a wide range of phenomena in mathematical language, focusing in detail on ambiguity in the symbolic mathematics. His syntactic analysis is based on a context-free grammar and semantics is modelled in a variant of Discourse Representation Theory modified for the language of mathematics. A formal type system is developed to account for ambiguity in the mixed, symbolic and natural language. Ganesalingam’s ultimate goal is to ‘‘build programs that do mathematics in the same way as humans do’’. Our goals in this thesis are, by comparison, much more modest and, of course, practically-driven. Two comments are made in relation to our work (Ganesalingam, 2009, p. 23): ‘‘The material produced by [users with ‘little to fair mathematical knowledge’] is not related to the formal dialect of mathematics’’ – as we will show, students’ mathematical language exhibits phenomena found in textbooks as well as a range of other language phenomena – and ‘‘[(Wolska and Kruijff-Korbayová, 2004a)] treats material in German, whereas we focus exclusively on English’’, which seems to suggest that the language phenomena found in German might be substantially different from those found in English. We translate our German examples preserving the syntax and semantics as closely as possible, in order to illustrate the cross-linguistic nature of the language phenomena found in our data. Neither Fox’ nor Ganesalingam’s analyses appear to be actually implemented.

1.3.3 Processing natural language proofs

Attempts at computational processing of informal natural language proofs have been so far based on constructed examples or restricted to short exemplary discourses. Ranta (1994, 1995b, 1996) analyses mathematical language in terms of Martin–Löf's type theory and in subsequent work builds a proof editor with natural language input based on a formalisation in the Grammatical Framework (Ranta, 1995a). The final analysis in (Hallgren and Ranta, 2000) appears to be oriented towards building appropriate type representations based on the input to the proof editor, rather than towards principled account of linguistic phenomena. A type-theoretical approach motivated by similar goals to Hallgren and Ranta's is also presented in (Callaghan and Luo, 1997).

Baur's (1999) approach is based on the LKB system (Copestake, 2001). Parsing is performed with a Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994) adapted for mathematical language, with λ-DRT based semantic construction (Bos et al., 1994). Basic phenomena found in an example proof from Chapter 2 of (Bartle and Sherbert, 1982) are addressed.

Likewise, Zinn (2004, 2006) uses exemplary proofs from Hardy and Wright's *Introduction to the theory of numbers* (1971) to illustrate his approach. Zinn claims that ''[t]he syntactic constructions of informal mathematical discourse are relatively easy, stylised or formulaic and more or less in line with English grammar rules'' and refers to Trzeciak's (1995) collection of ''standard phrases'' for mathematical texts noting that ''most mathematical arguments could be expressed by instantiating and combining these textual components'' (Zinn, 2004, p. 56). His linguistic analysis is partly guided by rules of good writing style in mathematics, such as those in (Knuth et al., 1989). More detailed analysis is dedicated to anaphora and conditionals (Zinn, 2004, p. 69). Computational processing is based on van Eijck and Kamp's λ-DRT (van Eijck and Kamp, 1997). As Ganesalingam observes, it often lacks generalisation (it assumes, for instance, that all mathematical constants ('1', '2', '3', etc.) are explicitly modelled in the lexicon), however, it appears that Ganesalingam is not right in claiming that embedded symbolic expressions should not be accessible for reference, as Zinn's account predicts (Ganesalingam, 2009, p. 20); consider '2' being accessible in ''$2 + 15$ is prime''. We will return to this when we discuss indirect anaphora in Section 3.2.2.5.

Natho (2005) and the TU Berlin Zentrum für Multimedia in Lehre und Forschung (MuLF) group developed mArachna, a language processing system for extracting knowledge from mathematical text.[12] The language addressed is German, therefore we review the approach in somewhat more detail.

[12] Between February 2005 (the time of publication of Natho's thesis) and 2008 around 20 articles related to mArachna have been published by MuLF researchers (`http://eprints.mulf.tu-berlin.de` [Accessed: 2012]). A system based on phrase structure grammar was presented in the articles from 2005 and 2006; a system based on HPSG in the articles from 2007 and 2008. Because the conceptual design and the actual text within the two sets of mArachna publications largely overlap, we will cite only one paper in which the information we refer to can be found.

Table 1.2: Natho's language structures in mathematical texts

Structure type	Template	Meaning
Implication	[VERB1] A, (so/dann) [VERB2] B	$A \Rightarrow B$
	wenn A [SEIN/GELTEN], dann [SEIN/GELTEN] B	
	wenn A [GELTEN], dann [GELTEN] B	
	falls A [VERB], dann [VERB] B	
	B [VERB] (nur/höchstens) dann, wenn A	
	A ist hinreichend für B	$A \Rightarrow B$
	dies ist hinreichend für B	
	dies ist eine hinreichende Bedingung für B	
	A ist notwendig für B	$B \Rightarrow A$
	eine notwendige Bedingung dafür ist B	
	aus A folgt B	$A \Rightarrow B$
	A dies hat zu Folge, dass / man kann folgern, dass B	
	A folglich [VERB] B	
	A, dies impliziert B	
	A, daraus ergibt sich / daraus erhalten wir B	
	A, das bedeutet B	
Equivalence	A ist äquivalent zu/gleichbedeutend mit B	$A \Leftrightarrow B$
	A [gelten] genau dann, wenn B [gelten]	
	A [gelten] dann und nur dann, wenn B [gelten]	
	A [sein] hinreichend und notwendig für B	
Quantifier	Für alle/jedes/ein beliebiges x...	$\forall x$...
	Jedes/zu jedem x...	
	Alle x...	
	Sei x beliebig...	
	Es gibt ein x...	$\exists x$...
	Für ein geeignetes x gilt...	
	[SEIN/HABEN] ein x...	
Set theoretic	...ist Element von...	...$\in$...
	...kommt in...vor	
Assumption	(es) sei... / ist...	assumption:...
	für...gegeben (ist/sei)...	
	es gelte...	

Like Zinn, Natho claims that the range of linguistic constructions in mathematical language is limited (Natho, 2005, p. 108), sentences with logical operators and quantifiers are in most cases simple, short, clear, and easily comprehensible, syntactic and semantic ambiguities are avoided through the use of phrasings with fixed meaning; Table 1.2 (p. 49) shows typical constructions (Natho, 2005, Section 3.3.3). The number of verbs used in mathematics ‘‘appears to be small’’. The most frequent verbs in German are: ‘‘sein’’ (*be*), ‘‘heißen’’ (*be called/named*), ‘‘existieren’’ (*exist*), ‘‘geben’’ (*be given*; corresponds to the English existential construction *there is/are*), and ‘‘folgen’’ (*follow*). Natho claims that mathematical expressions exhibit specific syntax that is in principle simple yet ‘‘*incompatible with the syntactic structures of natural language*’’ (2005, p. 108; emphasis added).[13] Some of the analyses are not linguistically informed; for example, on page 129, phrases ‘‘absolut konvergent’’ and ‘‘linear unabhängig’’ are given as examples of phrases with two adjectives one after the other (‘‘zwei Adjektive hintereinander angeordnet’’).

Linguistic analysis in mArachna is based on a four-level ‘‘linguistic classification scheme’’. The *Sentence Level* and the *Word and Symbol Level* describe ‘‘the characteristic sentence structures, which are commonly found in mathematical texts’’ and ‘‘[schematize] single symbols, words, and their relations between each other’’ (Grottke et al., 2005a). Assumptions, propositions, and properties are identified based on ‘‘stereotypical syntactic constructs and common phrases within their sentence structure’’ (Grottke et al., 2005c). This approach is similar to Zinn's, however, the authors of mArachna seem to be unaware of this work. Mathematical expressions are ‘‘generally’’ separated from the surrounding text and represented in the MathML[14] format. The authors do not say in which cases what other procedures are applied. Simple mathematical expressions within text are ‘‘replaced by placeholders’’ (Jeschke et al., 2008) while ‘‘some simple symbols and equations can be replaced by natural text elements’’ (Natho et al., 2008). Unfortunately, there are no examples to illustrate this substitution. mArachna does not seem to account for syntactic and semantic interactions between the two language modes, mathematical expressions and natural language, however, the authors plan to extended it to process ‘‘more complex formulae’’ using ‘‘syntactical analysis similar to those used in computer algebra systems in combination with contextual grammars (e.g. Montague grammars) to correlate the information given in a formula with information already provided in the surrounding natural

[13] ‘‘Mathematische Symbolfolgen wie z.B. Formeln weisen eine Ihnen eigene Syntax auf, die zwar prinzipiell einfach ist und auch durch die Prädikatenlogik strukturiert wird, jedoch nicht kompatibel zur syntaktischen Struktur der natürlichsprachlichen Texte ist.’’

[14] `http://www.w3.org/MathML`; see also the section on annotation languages further in this chapter.

language text'' (Jeschke et al., 2007a,b). The authors suggest that ''[u]sing this approach should enable mArachna to integrate formulae and their informational content in the network created by the analysis of the natural language text''. Unfortunately, there is no specific information as to the use of Montague grammars to process mixed language and the provided description is too vague to draw conclusions and let alone to compare the method with our proposal.

Chomskian analysis of an example sentence is shown in (Natho, 2005).[15] Unfortunately, details of processing are not elaborated. A later approach uses TRALE (Müller, 1999), a Head-Driven Phrase Structure Grammar parser which ''has been extended by expanding the dictionary and grammar to include the specifics of mathematical language'' (Natho et al., 2008). The output provides a ''comprehensive syntactic and even some partial semantic information about each sentence''. Neither details on the HPSG resources nor examples of lexical entries are provided. TRALE's output ''is transformed into an abstract syntax tree, symbolising the structure of the analysed sentence''.

Because TRALE cannot process mathematical expressions, formulas and terms must be processed separately from natural language, however, neither processing complex mathematical expressions nor interpretation of mathematical expressions within the surrounding natural language context has been implemented (Natho et al., 2008). Jeschke et al. (2008) mention that the symbols and equations (at this point unanalysed) are ''tagged with an identity number, and treated like a noun in the NLP analysis'', that is, the same way as in the approach based on phrase-structure grammar; see (Grottke et al., 2005b).

The semantic analysis in the HPSG-based system ''is implemented in the form of embedded JavaScript interpreter'' which categorises the syntax trees ''according to typical structures characteristic for specific mathematical entities and semantic constructs'' (Natho et al., 2008). The trees are subsequently transformed into triple structures using ''external JavaScript rules [which] map typical mathematical language constructs onto the corresponding basic mathematical concepts (e.g. proposition, assumption, definition of a term, etc.)'' The triples are annotated with the information about ''the context within the original document'' and about their ''classification within the context of the final OWL documents... [that is] [f]or each element of the triples it has to be decided if they represent OWL classes or individuals – complicating the semantic analysis'' (*ibid.*). Due to the general vagueness of the descriptions it is hard to relate the approach to other approaches and to our proposal.

[15] ''Durch Transformationen wird der Satz in seine Einzelbestandteile zerlegt, Phrasen ersetzt und Verben umsortiert. Dadurch entstehen strukturierte Satzbausteine, die syntaktisch nach dem Chomsky-Modell analysiert werden können'' (Natho, 2005, p. 126).

1.3.4 Controlled (natural) languages for proofs

SAD (Verchinine et al., 2007), Naproche[16] and MathNat (Humayoun and Raffalli, 2010) are examples of interactive proof construction systems based on controlled natural languages (CNL) which allow users to enter proof steps using a language that is close to natural language. CNL-based approaches assume that the vocabulary and the range of syntactic structures is a predefined subset of a natural language. Semantic interpretation can thus be restricted to processing the specific constructions allowed by the CNL grammar. The above-mentioned CNLs, however, do not offer a lot of flexibility of linguistic expression, for instance, as far as embedding symbolic mathematical expressions within natural language or using referring expressions are concerned. Humayoun and Raffalli claim to resolve certain types of referring expressions within their MathNat system, however, the reference phenomena addressed appear to be based on an exemplary constructed discourses rather than on real data and they are of course restricted to the scope of their CNL. Therefore, it is not clear how well the reference resolution methods would perform on a larger scale.[17]

Isar, of the Isabelle/Isar framework, while not a CNL, is a formal proof language designed for human readability (Wenzel, 2007). The MIZAR language (Trybulec, 1978; Rudnicki, 1992) and the SAD's ForTheL (Vershinin and Paskevich, 2000) were designed with the same motivation. While flexible in the sense that they enable defining new language constructs which can be immediately used within the constructed discourse, the price is that the discourses need to be self-contained, in that the vocabulary – all lexemes and their semantic interpretations – needs to be formally specified in the proof document. Since in this thesis we are interested in natural language proofs we will not elaborate on controlled natural languages any further.

1.3.5 Proof annotation languages

In parallel to computational processing, manual annotation of proofs has been proposed as a methodology for studying mathematical discourse or as part of semi-automated processing. Because manual annotation is not an approach that we can consider in a practical scenario of tutoring we discuss markup languages for mathematics only briefly for the sake of completeness.

General languages for annotating mathematics Several markup languages for mathematical documents have been developed for the purpose of

[16] http://www.naproche.net

[17] MathNat is a successor of the DemoNat project whose goal was to develop a natural language-based proof tutor for French; see http://wiki.loria.fr/wiki/Demonat [Accessed: 2006].

displaying mathematics in web browsers or in the context of the semantic web. MathML and OpenMath are languages for representing the structure and semantics of mathematical notation.[18] OMDoc[19] (Kohlhase, 2006) is a general semantics-oriented markup for mathematical knowledge which extends OpenMath to entire discourses. sTeX[20] markup, developed by the OMDoc's author, enables semantic annotation of mathematics directly within LaTeX documents.

Proof Markup Language ProofML (Schröder and Koepke, 2003) is a linguistically motivated markup for proofs focusing on sentence and discourse-level semantic phenomena, such as logical structure (the scope of the premises and consequents markup), linguistic quantification devices (quantificational determiner, restrictor, and scope markup), distributive and collective readings of plurals, and ellipsis. While semantically annotated mathematical documents would be extremely valuable for studying the relations between the linguistic and logical structure of proofs, we are not aware of any ProofML-annotated corpora (other than the three-sentence proof in the paper's appendix).

MathLang The purpose of MathLang (Kamareddine and Wells, 2001, 2008) is to enable semi-automatic computerisation of mathematics written in ''common mathematical language'' – the language and style in which mathematicians routinely write – into *any* language of *any* proof checker. The assumption is that a scientist, while working on a mathematical paper, would annotate his/her own document by explicitly identifying and labelling segments of text using MathLang markup. Unlike ProofML, MathLang distinguishes different levels (''aspects'') of annotation granularity – the Core Grammar aspect (CGa), the Text and Symbol aspect (TSa), and the Document Rhetorical aspect (DRa) – which from a computational linguistics point of view correspond to the following processing steps: grammatical analysis, analysis of symbolic mathematical expressions, lexical and type semantic analysis, and discourse analysis. Here we only briefly outline certain peculiarities of the CGa and the TSa aspects.

The CGa is a kind of type system inspired by Weak Type Theory (Nederpelt and Kamareddine, 2001; Kamareddine and Nederpelt, 2004) and de Bruijn's mathematical vernacular. Its markup, shown in Table 1.3, is partly linguistically and partly domain-motivated and corresponds to the annotation of grammatical categories and certain types of discourse segment categories in text. Each colour-coded *annotation box* is annotated with semantics. The semantics of

[18] `http://www.w3.org/Math`, `http://www.openmath.org` [Accessed: 2006]

[19] `http://www.omdoc.org` [Accessed: 2006]

[20] `https://trac.kwarc.info/sTeX` [Accessed: 2006]

a coloured box is indicated in the form of ''interpretation attributes'' which symbolically represent the domain interpretation of the boxed text fragment.

A CGa analysis of the sentence ''There is an element 0 in R such that $a + 0 = a$'' is shown in Figure 1.3.[21] At the semantic level, the blue *term* boxes, are tagged with attributes `0` and `plus`. The *statement* box is tagged `eq`. The complete CGa annotation of the sentence is shown in Figure 1.4. A text segment colour-coded in this manner can be rewritten in MathLang's abstract syntax (Kamareddine et al., 2006) by reading off the annotations:

```
{ 0 : R;   eq ( plus ( a, 0 ), a ); };
```

The link between the ''common mathematical language'' and formal interpretation is established by the TSa level and facilitated by *souring* annotations which, unlike the CGa categories, are somewhat less linguistically-informed.

Kamareddine et al. (2007a) observed that in mathematics the surface language does not always directly ''match'' the intended domain interpretation. As an illustration of a simple phenomenon which motivated *souring*, consider the well known convention of chaining equations:

$$0 + a0 = a0 = a(0 + 0) = a0 + a0$$

Its obvious interpretation is a conjunction with some terms duplicated (shared):

$$0 + a0 = a0 \wedge a0 = a(0 + 0) \wedge a(0 + 0) = a0 + a0$$

The purpose of souring is to recover the intended meaning, while preserving the imprecise surface realisation in expressions such as above.[22] In line with MathLang philosophy, souring is a tagging task. ''Sour bits'' are added to the text by means of special boxed annotations with a thick frame and a distinct colour. The authors claim that *re-ordering*, *sharing/chaining*, and *list manipulation* transformations are required in order to handle certain phenomena. From a linguistic point of view, these correspond to linearisation, aggregation, and certain types of ellipsis phenomena in natural language.

The MathLang souring transformations with examples are illustrated in Table 1.4 (from (Lamar, 2011, p. 78)). A reordering transformation is performed when the linear order of words or symbols does not agree with the order defined in the formal language. As an example of this phenomenon Lamar notes that

[21] Examples from (Kamareddine et al., 2007a,b, n.d.).

[22] The term *souring* was invented by analogy with the notion of *syntax sugaring* in programming languages. ''Syntactic sugar'' is added to programming languages in order to make their syntax easier to read and write for humans. Here, the opposite is needed: For the purpose of computerisation and formalisation, the content which is not realised on the surface must be restored. Therefore, one can think of the common mathematical language as ''sweet'' and of the formalisation language as ''sour'' (Kamareddine et al., 2007a).

Table 1.3: MathLang's Core Grammar categories

Category	Description	Example
term	a mathematical object	``a+b'', ``an additive identity 0'', ``$\sqrt{2}$''
set	a collection of objects	``$\mathbb{N}$''
noun	a family of objects which share common characteristics	``ring'', ``number''
adjective	a modifier which constructs new **nouns**; for instance, by refining old ones	``Abelian'', ``even''
statement	a unit which has a truth value, describe mathematical properties	``$a = a$'', ``P lies between Q and R''
declaration	a unit which gives a signature to a new **term**, **set**, **noun**, **adjective**, or **statement**	``Addition is denoted $a + b$''
definition	a unit which defines new symbols	``A ring is...'', ``A number p is prime whenever...''
context	a unit which sets preliminaries prior to a **step**; for instance, a **statement**, a **declaration** or a **definition** restricted to a specific part of a document	``Given a ring R,...''
step	a **statement**, a **declaration** or a **definition**, a succession/sequence thereof (i.e. a **phrase/block**), or a **context**	``We have...''

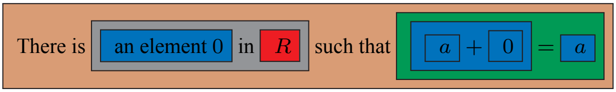


Figure 1.3: An example MathLang CGa analysis

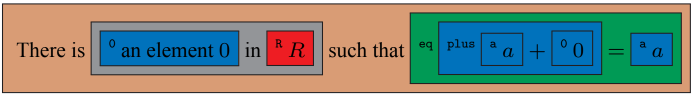


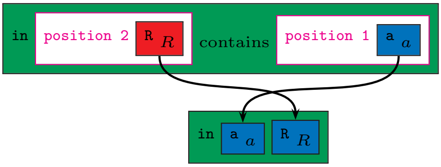
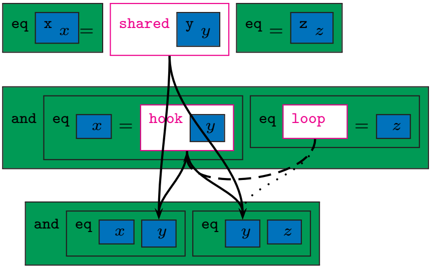
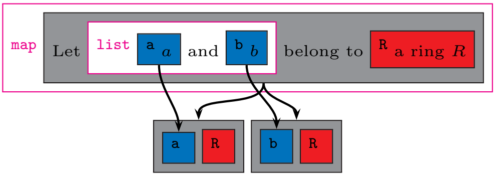
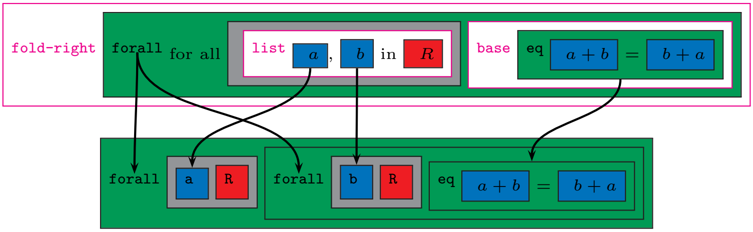
Figure 1.4: MathLang CGa analysis with semantic annotations

the formal set membership notation and the linearisation of the prepositional phrase with ''in'', on the one hand, and, on the other hand, the order of arguments of the verb ''contain'', whose intended interpretation is that of set membership, do not ''match'': We write and say $a \in R$ and ''a in R'', but ''R contains a''. Thus, he suggests, in the latter case the arguments must be reordered so that the intended representation, `in(a,R)`, can be obtained. To this end, the clause ''R contains a'' is annotated with *position* information and this annotation is used to transform it to the formal representation, uniform to all expressions whose intended meaning is that of set membership. *Shared* and *loop–hook* tags are used when a segment has to be duplicated. A typical example involves the previously mentioned chaining equations. *Folding* and *mapping* annotations are used in list contexts to repeat a segment for each element of a list when the intended repetition was suppressed or elided. A typical example is quantification over multiple variables, that is, clauses such as ''for all x, y, z,...'' In the formal language the quantifier is recovered (''unfolded'') for each bound variable. This is achieved by repeating the appropriate annotated segment. While, admittedly, aggregation and ellipsis resolution do require that a discourse-level interpretation process recovers the underlying semantics, for instance, by means of a coindexing mechanism, in a way analogous to the effect of the souring transformations, clearly, a linguistically informed grammar and a principled syntax–semantics interface would enable analysis without the reordering transformations.

1.4 Discussion

As the second part of this chapter shows, processing natural language proofs has been an ''ongoing research project'' for decades. In fact, processing students' natural language proofs had been done previously and on a large scale (at Stanford). Processing mathematical prose is not a new direction in natural language processing either. So is the problem solved? Far from it. Although several approaches to computational processing of mathematical discourse have been proposed, it appears that most of the recent work on the natural language of mathematics has focused on theoretical models (Fox, Ganesalingam) whereas the coverage of the implemented approaches have been anecdotal. Baur and Zinn process only a small set of sentences. Baur models 3 proofs; around 30 sentences in total, of which some have the same syntactic structure. Zinn ''[is] only aware of [his system] being able to completely process the two example constructions in [his] ch. 7'' (Zinn, 2004, p. 199). That's 9 sentences. mArachna appears to exist as a proof of concept implementation that demonstrates the feasibility of the approach

Table 1.4: MathLang *souring* transformations

Phenomenon *MathLang terminology*	Boxed annotation	Examples
Linearisation *Re-ordering*	`position` i	 in position 2 R R contains position 1 a a in a a R R
Aggregation *Sharing/chaining*	`shared` `hook-loop`	 eq x x = shared y y eq = z z and eq x = hook y eq loop = z and eq x y eq y z
Ellipsis *List manipulation*	`map`, `fold-right`, `fold-left`, `base`, `list`	 map Let list a a and b b belong to R a ring R a R b R
		 fold-right forall for all list a, b in R base eq a + b = b + a forall a R forall b R eq a + b = b + a

''[f]or selected text elements'' (Jeschke et al., 2007b,a). The running example of a definition of a group consists of 5 sentences. The descriptions of mArachna are too vague, therefore we are not convinced of the scalability of the approach.

Unlike Zinn's approach which relies on a tight correspondence between the representations produced by linguistic analysis and the representation used for reasoning, we argue that an architecture for processing mathematical discourse and an interpretation strategy for processing mathematical language should be designed in a modular fashion, rather than be coupled with a prover, in order to be flexible enough to support the different application scenarios outlined in the beginning of this chapter. In particular, the semantic representation should be independent of the reasoner system's input representation, so that the functionality is not bound to a specific deduction system. The interpreted linguistic meaning representation which we propose as the semantic output representation does possess this property.

In practice, with the exception of Ranta's GF, no reusable grammar resources for mathematical language appear to exist. We do not use GF for two reasons: First, we choose Combinatory Categorial Grammar because it is an expressive grammar formalism with a perspicuous syntax–semantics interface, which enables modelling complex linguistic phenomena in a transparent way (Steedman, 2000; Baldridge, 2002); for instance, complex coordination phenomena, notoriously difficult for grammar formalisms, or word order phenomena. Moreover, the parser implementation which we use produces logical forms which can encode domain-independent linguistic meaning, such as those we would like to obtain, in terms of dependency semantics. Our approach is related to Ranta's in the sense that Categorial Grammar is also a kind of type system. Our grammar's design is, however, linguistically-motivated and, as we will show in Chapter 7, it provides good linguistic generalisations. Second, the concrete grammars in GF appear to exist for a set of constructed examples. In this work, we wanted to model actually recurring phenomena based on authentic linguistic data. To this end, we collected a corpus of students' interactions with a simulated system, in order to investigate language phenomena naturally occurring in this discourse genre. The following chapter motivates the choice of data acquisition methodology and outlines the setup of the data collection experiments.

Chapter 2

Corpus acquisition

This chapter summarises two corpus collection experiments conducted to acquire authentic data on pedagogical, mathematical, and linguistic aspects of proofs constructed by students in naturalistic computer-mediated tutorial dialogue interactions. The first experiment was the first, to our knowledge, medium-scale effort to collect empirical data on human–computer tutorial dialogues on mathematical proofs, on the use of natural language in proof tutoring, and on dialogue phenomena specific to such interactions. The proof exercises in this experiment concerned naïve set theory. Building on the insights from the first experiment we conducted another experiment on proofs in binary relations. This time we were interested in two issues: first, in the language production – in particular, factors that influence the character of the language that students use – and second, in the issue of proofs' granularity – argumentative complexity – specifically, in the differences between granularity appropriate from a pedagogical point of view and that required by automated deduction systems. Before summarising the experiments and presenting the corpora, we discuss the motivation for collecting new data, rather than using existing data (such as textbook proofs or proofs extracted from scientific publications). After summarising alternative dialogue research methods, we motivate our choice of methodology, a system simulation.

2.1 Motivation

Proofs are central to doing and knowing mathematics and omnipresent in mathematical discourse. The language of informal proofs can be studied based on the enormous body of printed and electronic mathematical publications. In the introductory chapter, we already mentioned Baur's and Zinn's work on computational processing of textbook proofs based on isolated examples from specific texts (see p. 47). Since we are motivated by the same ultimate goal – automating the linguistic interpretation of proofs – a question arises whether our language processing method could be based on the study of the same kind of data. Although this idea seems rather attractive, mainly due to the ease of access to research material, there are reasons why textbook proofs alone should not guide the computational analysis when the aim is to process (i) students' proofs, (ii) constructed in a dialogue interaction, and (iii) with a computer.

Mathematical textbooks are written by expert mathematicians. The ''writing styles'' of experts differ from the styles of novices in maths. They even differ among mathematicians themselves; proofs of the same theorem presented by different authors may be entirely different even if the underlying proof ''idea'' and structure are the same. Even the same mathematician might produce different proofs depending on to whom the proof is addressed:

> [...] the style of writing need not be the same when you address yourself to an expert or to a beginner. [...] For research monographs, I would [...] consider as satisfactory [...] allowing some looseness in the general organisation, the skipping of a lot of proofs or comments which are trivial for experts, etc. On the contrary, when it comes to textbooks aimed at beginners, I am entirely in agreement with Halmos regarding the necessity of a very tight organisation, and I would even go beyond him with regard to the ''dotting of the i's''; this may well be annoying to the cognoscenti, but sometimes it will prevent the student from entertaining completely false ideas, simply because it has not been pointed out that they are absurd.
> (Dieudonné in (Steenrod et al., 1973))

A common property that expert mathematicians' proofs should share – aside from validity which in the case of textbooks we take for granted – is that a proof should be *convincing* from an argumentative point of view: it should be presented in such way and with such level of detail that a reader to whom it is addressed can accept it as a proof of the given proposition. Again, educational material, such as textbooks, requires special attention to detail:

> [...] in a research monograph a great many things may remain unsaid, since one expects the expert reader to be able to fill in the gaps; [...however, even in that case...] you may very often skip a single line of a proof, but never two consecutive ones. For textbooks, on the contrary, [...] all the details must be filled with only the exception of the completely trivial ones. (Dieudonné, *ibid.*)

By contrast, proofs produced by novices in a learning setting often differ from those published in textbooks in that they are invalid (use invalid inferences or state false propositions), incomplete, or otherwise inappropriate from a pedagogical point of view (use inappropriate representations, omit necessary parts of argumentation) or contain formal inaccuracies (Selden and Selden, 2003). Proofs constructed in a dialogic tutoring interaction may moreover contain discarded unsuccessful starts, false conjectures and conclusions corrected in the course of tutoring either by the student or the tutor, restarts, or changes of strategy. These kinds of discourse disfluencies are typical of dialogue in a pedagogical setting and are not often found in written narrative texts.[1] Doing proofs in an interaction with a tutor has a character of an argumentative dialogue in which the learner has to provide arguments to show that, on the one hand, a proposition in question holds or does not hold, and, on the other hand, that he has a deep understanding of the mathematical objects involved, the relations among them, of the method employed to find the proof, and of the theorem's mathematical implications, rather than that he can merely state a theorem or a definition. Thus, analysis of experts' proofs would omit proof aspects typical of learner presentations and of dialogic interaction. Textbook proofs can give a general idea of the expectations of the given textbook's author as to how rigorous students' proofs should be. However, since our goal is to understand and model students' proofs, we need a corpus of students' proofs.

Interpretive studies into proving and problem solving often use research designs that involve collecting corpora of students' problem solving and interaction with tutors in the classroom or out-of-school laboratory settings. Common designs in qualitative research include clinical methods, teaching experiments, and classroom research (Kelly and Lesh, 2000). Data collection techniques include open-ended surveys, structured task-based interviews, stimulated recall interviews, think-aloud protocols, field notes and video-/audio-taping of classroom activities (*ibid.*). Most studies involve interactions between students and human tutors or between peer students. While some studies do report on educational uses of computer programs such as Computer Algebra Systems (Schneider, 2000; Heid and Edwards, 2001), proof tutor systems (Suppes and Morningstar, 1972; Suppes, 1981; Goldson et al., 1993; Scheines and Sieg, 1994; Abel et al., 2001; Borak and Zalewska, 2007) or web-based environments for learning mathematics, also learning to prove (Ravaglia et al., 1999a,b; Melis et al., 2001, 2006; Hendriks et al., 2010), at the time this project began there was no available data on natural language computer-based tutoring of proofs. Therefore, in order to learn about the characteristics of tutorial dialogues on proofs, in particular, about the students' language, we performed a series of controlled experiments to collect data in our target scenario.

In the reminder of this chapter, after discussing methodological considerations, we present an overview of the data collection experiments. The experimental design and an overview of the data collected in the first experiment were summarised previously in (Benzmüller et al., 2003a,b; Wolska et al., 2004b) and in the second experiment in (Benzmüller et al., 2006; Benzmüller et al., 2006; Wolska and Kruijff-Korbayová, 2006a).

[1] Lakatos' *Proofs and refutations* is of course an example of a notable exception.

2.2 Methodological considerations

The choice of research methodology adopted to investigate the structure and properties of discourse, depends, among others, on the availability of prior data in the area of interest and on the ultimate research setting: theoretical (basic research) vs. practical (applied). Early foundational studies in pragmatics and conversation analysis, such as those of Austin, Searle, and Grice, whose goal was to construct theoretical models of human communication, were predominantly based on introspection or on studies of human–human dialogues. In applied research on dialogue systems, the adopted methodology should facilitate computational modelling and identification of requirements on the functionality of the systems' subcomponents. Functional and technical requirements can be determined using several methodologies, including studying similar systems through literature research, analysis of existing data, conducting user interviews to elicit knowledge on the domain and task, by field-study observations of humans performing the task in question, by rapid prototyping, or partial and full-scale simulations (McTear, 2004). In dialogue systems design, two of the most commonly employed methods are analysis of large collections of (transcribed) human–human dialogues and system simulations.

The motivation for choosing the research methodology in this project was two-fold: First, the goal was to obtain a corpus of students' dialogues on proofs. Second, it was to identify functionality requirements for subcomponents of a prototype system, based on the analysis the collected data. Especially relevant for the work presented in this thesis were the requirements on the input interpretation module. Below we briefly discuss frequently applied research methodologies, and then present the general design of a Wizard-of-Oz study, the experimental paradigm we adopted.

Related corpora As the fields of speech and dialogue research mature and dialogue systems, also spoken dialogue systems, slowly become commercial reality rather than purely academic research (Dahl, 2004; McTear, 2004) corpora collected in academic projects become available to the dialogue research community through organised initiatives, such as the LDC or SIGdial.[2] Similarly, as deployed Intelligent Tutoring Systems actually enter classrooms (Anderson et al., 1995; Koedinger et al., 1997; VanLehn et al., 2005), samples of interactions become available. However, most existing tutorial dialogue corpora, do not concern formal domains such as ours. Notable exceptions are the data related to Ms. Lindquist (Croteau et al., 2004), PACT (Popescu and Koedinger, 2000), and AGT (Matsuda and VanLehn, 2005), but the interfaces of those

[2] `http://www.ldc.org`, `http://www.sigdial.org` [Accessed: 2006]

systems support prescripted menu-based user input or only short natural language responses, thus the interactions with those systems do not represent the kind of flexibility in the use of natural language and dialogue that we aim at.[3]

Analysis of human–human interaction Study of human–human interaction is an established methodology in dialogue research employed to inform theoretical modelling and computational implementation of discourse and dialogue processes; see (Grosz, 1978; Reichman, 1985; Clark, 1996) to mention just a few. Non-interventionist research, such as observation of student–teacher interactions in a naturalistic classroom setting or field studies of human tutoring, is also commonly employed in the mathematics education community (Kelly and Lesh, 2000). When specific research questions are asked, controlled experiments, for instance, one-to-one semi-structured clinical interviews, are conducted (Ginsburg, 1981). Data analysis in those settings is based on transcripts of audio and/or video recordings of student talk (with or without a teacher), debriefing questionnaires, and/or post-experiment interviews with the subjects conducted by the experimenter. Observations of human tutoring have also been used in Intelligent Tutoring Systems research to identify those characteristics of human tutoring that make tutor-assisted instruction produce a larger difference in learning gains than classroom instruction (Bloom, 1984) and to investigate the weaknesses and limitations of the state-of-the-art automated tutoring; see, for instance, (Merrill et al., 1992; Aleven and Koedinger, 2000; Heffernan and Koedinger, 2000; Person and Graesser, 2003).

While studying human tutoring in complex problem-solving tasks, such as proofs, is interesting in itself, empirical evidence indicates that humans behave differently when they interact with other humans than when they interact with machines (Richards and Underwood, 1984; Morel, 1989; Fraser and Gilbert, 1991; Dahlbäck et al., 1993; Yankelovich et al., 1995; Bernsen et al., 1998; Pirker et al., 1999; Shechtman and Horowitz, 2003). Most of the studies cited here concern spoken dialogue. Richards and Underwood (1984) and Morel (1989), for instance, found that, aside from speaking more slowly and clearly, in man–machine interaction humans use a more restricted language, both in terms of syntax and vocabulary, ask fewer questions, and avoid complex or potentially ambiguous anaphoric references. In a study on tutoring, Rosé and

[3] A corpus of learner interactions with an ITS for teaching calculus has been collected within the LeActiveMath project (`http://www.leactivemath.org` [Accessed: 2006]). However, the LeActiveMath corpus is not publicly available. DemoNat (`http://wiki.loria.fr/wiki/Demonat` [Accessed: 2006]) is another project on automated natural language tutoring of proofs. A sample of French dialogues obtained in simulated interactions has become available, however, the corpus is too small to make generalisations as to the properties of the discourse and as to what language phenomena occurring in French would also occur in other languages.

Torrey (2005) found that students contribute more self-explanation if they believe that they are interacting with a human than when they believe that they are interacting with a computer. Users also ‘‘align’’ with the system in terms of linguistic style; this phenomenon has been exploited in attempts to shape (or to a certain extent control) users' input (Leiser, 1989; Ringle and Halstead-Nussloch, 1989; Zoltan-Ford, 1991; Brennan and Ohaeri, 1994; Tomko and Rosenfeld, 2004). Thus, when performing experiments which involve unrestricted human–human interactions one has to bear in mind that the complexity of the obtained data might be greater, possibly even beyond the scope of a realistic computer-based scenario, than in an experiment in the target scenario involving a machine. This may in turn lead to specification of unrealistic functionality requirements and it may be difficult to formulate conclusions about how a corresponding man–machine interaction might look.

Rapid prototyping Rapid prototyping (McTear, 2004; Dahl, 2004) is a methodology typically employed in commercial systems if the task complexity allows the designers to build a system's subcomponents quickly by anticipating possible target interactions or by interviewing the prospective users about their expectations. A prototype system is an autonomous application which includes the core of the domain-relevant processing, which, however, may not have the full functionality of the final system; for example, the range of accepted user utterances or the linguistic variation in the generated output may be limited. Such a limited-functionality system may then be used in pilot usability tests to inform further development. Because of the complexity of our target task and the fact that little data exists on dialogue-based computer tutoring of proofs, early prototyping was not considered as a methodology to be adopted.

Partial and full-scale simulations When the complexity of the task scenario is considerable and there is no existing system with the anticipated functionality, a simulation may be conducted in order to collect data on how humans interact in the scenario in question. Aside from giving insight into language phenomena and interaction patterns, analysis of the obtained data can serve to lay out functionality requirements for the system's subcomponents. Simulations, often referred to as *Wizard-of-Oz experiments*, have been long employed in the human factors research, experimental psychology, usability engineering, and also dialogue systems (Gould et al., 1983; Kelley, 1984).[4]

[4] ‘‘Wizard-of-Oz’’ is an obvious reference to a character in the 1900 children's story *The Wonderful Wizard of Oz* by Baum, in which Oz, the terrible ruler of the Emerald City, turns out to be a marionette operated by a little old man behind a screen who pulls at strings to make the puppet's eyes and mouth open. The term was coined by Kelley. Another term he used was *OZ Paradigm*

The idea of a Wizard-of-Oz (WOz) experiment is that a human (the wizard) simulates the role of a hypothetical intelligent application in a laboratory setting by providing the system's responses to the experiment participants (the subjects/users). In the case of spoken interaction, the wizard, for instance, types responses on the keyboard and voice output is synthesised by a text-to-speech system. The subjects and the wizard are physically separated during the experiment to exclude communication outside the mediation interface. The experiment may be conducted with the subjects' prior knowledge of the simulation, however, in order to elicit natural behaviour, participants are often made to believe that they are interacting with a computer.[5] The decisive factors in adopting the WOz methodology for our studies were the following:

Authenticity of data The collected data is a believable sample of interactions in the target scenario in that the ''human factor'' causing differences between interactions with humans and machines is removed.

Affordability Building a simulation environment is typically easier and less costly than building a fully-fledged application or even a prototype. Simulation environments created in previous projects might be reused provided that the new setting is sufficiently similar to the one for which the original tool was developed and that the tool fulfils the requirements of the user interface in the new setting.[6]

Iterative design Kelley (1984) and later Fraser and Gilbert (1991) proposed a WOz-based multi-stage methodology of principled, empirically grounded iterative development of complex applications which comprises six steps of system development:

1. *Task analysis* The structure of the task is investigated;
2. *Deep structure development* Data access functions for the wizard are developed;
3. *First run of WOz (simulation)* The system is fully simulated;
4. *First-approximation processor* The corpus from the simulation phase is analysed and the first approximation of the input understanding subcomponent is developed;
5. *Second run of WOz (intervention)* The system is partially simulated: the component developed in step 4. is integrated into the

and *OZ* stood for ''Offline Zero'', a reference to the fact that the wizard interprets the input and responds in real time (see `http://musicman.net` [Accessed: 2006]). PNAMBIC (Pay No Attention to the Man Behind the Curtain) is another early name of the technique (Fraser and Gilbert, 1991).

[5] For ethical reasons, the deceit should be disclosed during debriefing after the experiment.

[6] A dedicated simulation tool enabling alternative methods for mathematical formula entry has been built for each experiment; for the motivation, see (Fiedler et al., 2004; Benzmüller et al., 2006).

simulation environment and the wizard simulates the remaining parts or intervenes, when necessary, to keep the dialogue flowing;

6. *Cross-validation* Final application testing.

Steps 4., 5., and 6. can be repeated in a cycle an arbitrary number of times.[7] In the process of successive iterations, the initial prototype is gradually refined and the application takes over the functions simulated by the wizard. Thus, partial simulations provide a way of empirically validating various aspects of an interaction model before its final validation in usability experiments with an implemented autonomous system.

User-centred empirical approach The main purpose of a WOz experiment is for researchers to observe the users' behaviour during interaction with the anticipated system and to evaluate the use and effectiveness of its interface, rather than the overall quality of the entire system. In this sense, the method is by design user-centred.

Support of exploratory research Studies of human–computer interaction can be carried out without a commitment to application development.

Since Gould and colleagues and Kelley, the WOz technique has been applied in a variety of settings and tasks and to address diverse research questions, also in (tutorial) dialogue systems research. Given the complexity of the tutoring domain and the benefits of an empirical design, we considered the WOz paradigm an appropriate methodology to achieve our initial goal of data collection. Two points about the WOz methodology have to be kept in mind though. A major problem in a real-time simulation involving a human substituting for a machine is the significant cognitive load on the wizard. The wizard must perform the following tasks in the shortest possible time while preserving consistency of responses and avoiding erroneous transmissions to the user: (1) intercept the input (this may involve just listening to the transmitted audio or reading text on a screen, but also, in the case of multi-modal input, pointing gestures and graphical events), (2) interpret it, (3) perform the problem-solving task (this may involve accessing information from a database or performing reasoning related to the current task state), and (4) generate a response. It is clear that the wizard's task is demanding and that flawless behaviour borders on impossible. Not surprisingly, a recurring observation reported in WOz studies is that the users found the simulated system slow. This is because wizards tend to pay attention to task-level precision and the quality of the output at the sacrifice of response-time. Some of the cognitive load

[7] Fraser and Gilbert's cycle is in essence the same: the *second or subsequent experimental phases* collapses this loop into one step; the *pre-experimental phase* corresponds to steps 1. and 2., the *first experimental phase* to step 3.

can be relieved by using a setup with an interface in which the wizard's GUI contains menus of precompiled responses (Dahlbäck and Jönsson, 1989) or by using a multi-wizard setup (Francony et al., 1992; Salber and Coutaz, 1993).[8]

The second issue that should be kept in mind is that unrestricted simulation, that is, one in which the subjects' and the wizards' behaviour is *not* intentionally constrained, be it by the interface (making it reflect a *realistic system's* implementation) or by interaction protocols (shaping the interaction to correspond to a *realistic system's* capabilities; in our case, computationally plausible semantic analysis, tutorial dialogue modelling, language generation, and reasoning), produces data which correspond to an *idealised system*, one with all the processing capacities of a human. To remedy this, the experimental setup can be designed in such way that it limits the interaction in certain aspects, so that it corresponds more closely to the anticipated realistic system. Our design decisions are summarised in the following sections.

2.3 Experimental setup

The basic philosophy underlying iterative incremental methodologies is to start simple and to increase complexity in subsequent iterations. Our design decisions reflect this philosophy in that in the technical aspects we favour the simpler over the more complex. The aspect of the interaction which we left unrestricted was the use of language. Below we briefly outline how we shaped interaction in the domain of mathematics (the interaction modality, constraints on the communication language, and the user interface for mathematical notation) and motivate the choice of the manipulated variables.

Mode of interaction In most real-life situations tutors communicate with students using spoken language. This is certainly true of one-on-one tutoring. At schools and universities, written communication is used in exams, homeworks, and nowadays also in student–tutor email exchanges. (An exception is remote schooling, where written communication may be used more often than in the typical scenario.) Mathematics is a special science in that in principle it can be communicated using its language of symbols, mathematical notation, alone. The informal language of mathematics consists of a mixture of natural language and symbolic notation.[9] Typically, in one-on-one tutoring, knowledge and

[8] In our second experiment, due to the difficulty that our tutors experienced in mentally processing long formulas under stress, we modified the experimental setup in order to make it possible for the tutors to start processing the subjects' input before it was submitted. We will return to this when we discuss the second experiment in Section 2.6.

[9] Mathematical language will be discussed in more detail in Chapter 3.

explanations are conveyed with speech, while writing serves those situations where visualisation or formality are needed. Thus, we need *both* languages to explain maths: justify the inferences in words and express mathematical facts (proof steps) either with words or formulas. The question is whether to a computer-based tutor we should speak or type.

Speech is the most natural form of human communication. It is also the preferred modality in computer-mediated task-oriented dialogues (Rudnicky, 1993; Allen et al., 1996). However, textual interaction has the advantage of easy access to the prior discourse history (Herring, 1999; Gergle et al., 2004), which is relevant in tutorial dialogues as it helps the student keep track of what he has learnt and which tasks he has solved. While there are a few spoken tutoring systems (Mostow and Aist, 2001; Schultz et al., 2003; Litman and Silliman, 2004), to date the majority of dialogue-based tutors operate in typewritten mode (Rosé and Freedman, 2000; Heffernan and Koedinger, 2002; Zinn et al., 2002; Michael et al., 2003).

Speech may be preferred by users because it is faster to produce, but speech is certainly harder for a machine and, especially with mathematics as the domain, adds complexity to the interface implementation. Whereas considerable progress has been made in Optical Character Recognition towards recognising handwritten mathematical expressions[10] and programs capable of speaking mathematical notation do exist,[11] interfaces which enable speech input for maths or which combine speech and writing are not common. Interestingly, the main question is not whether the state-of-the-art automatic speech recognition (ASR) systems are in general powerful enough to support recognition.[12] The more fundamental question is: How should we speak math... to a computer? Although seemingly trivial – since we ''speak math'' whenever we talk about math – there is more to the question than it appears. The math we speak is typically accompanied by symbolic notation; relevant groupings are indicated by pauses and changes of speech tempo. Typically, there is no access to these features of speech in off-the-shelf ASR systems.[13]

[10] Blostein and Grbavec (1997) give an overview; see also the InftyProject, its publications and references therein (`http://www.inftyproject.org` [Accessed: 2006]).

[11] Raman's ASTER system (1994; 1997; 1998) is probably best known; Design Science MathPlayer plug-in is another example.

[12] There is a caveat here: typically, interpretation grammars in commercial ASR are finite-state. A mathematical expression parser needs more expressive power because of recursive subexpression embedding (Fateman, n.d.b).

[13] Consider speaking this simple set expression: $A \cap (B \cup C)$. In English, you probably produce something along the lines of ''A intersection <pause> B union C'', with a marked pause after ''intersection'' and with ''B union C'' spoken faster as one chunk of information. For an ASR system, one would probably have to produce something along the lines of ''A intersection open parenthesis B union C close parenthesis''.

Moreover, if both spoken and written input is to be used (typed on the keyboard or handwritten with a stylus) synchronisation becomes an issue.[14]

But is learning influenced in any way by the modality in the first place? So far, there is no evidence. In a study which compared human–human and human–computer spoken and typed tutorial dialogues Litman and colleagues (2005) found that while spoken dialogue is more effective in that tasks are faster accomplished, the augmented, spoken interface brings no significant difference in the learning gain by comparison with typed input. Interestingly, speech recognition errors do not negatively affect learning either (Pon-Barry et al., 2004; Litman et al., 2005). Thus, the above-discussed issues, the lack of corpora of computer-based proof tutoring, and the exploratory nature of our study make the simpler typewritten modality an obvious choice.

Use of natural language Since our central research objective was to collect data on the use of language in *authentic* computer-mediated tutoring, that is, as it should be if a computer system could have all the reasoning capacities of a human, we did not restrict the wizards nor the subjects in their language use. In one experimental condition of the first experiment the wizard followed a specific tutoring protocol which restricted his interaction and his use of language. The language production of the subjects and the wizards was otherwise not constrained in the other conditions and in the second experiment. Our goal was to find out how the participants cope with the need for natural language and mathematical expressions in proofs (given the limitations of the typewritten setup and the lack of spoken communication) and what language phenomena emerge as a consequence; for instance, whether the language turns out to be simple with little ambiguity, like in the experiments of Richards and Underwood (1984) or Morel (1989), and if not, whether the resulting language would have such *complexity* and *diversity* that the coverage of a parsing grammar in a prototype system would be poor.[15]

User interface for mathematical notation User interface design is one of the crucial elements in achieving natural, efficient communication with a computer. Plausible options for mathematical notation which do not involve speech, include: typing on a keyboard (mathematical expressions will typically

[14] For further issues in combining speech and writing in interfaces for mathematics, for an answer to the question of how we can and *should* speak math, and a description of a system prototype, see (Fateman, n.d.b). *Math Speak & Write* (Guy et al., 2004) and TalkMaths (Wigmore et al., 2010) are other examples of experimental systems.

[15] We attempt to answer these questions in Chapter 3 and Chapter 4, respectively. In Chapter 7 we evaluate the coverage of implemented parsing grammars based on the collected corpora.

have annotation or markup; as in LaTeX), GUI buttons, structured editors (as in EPGY TPE's ProofEd (McMath et al., 2001) or MathsTiles (Billingsley and Robinson, 2007)), or – the most complex alternative – handwriting[16]

The advantage of structured editors is that they provide templates for mathematical notational constructs and can internally encode the information on their valid types making immediate validation and diagnosis of semantic or syntactic errors possible. A structured editor area in a GUI, however, explicitly separates the natural language from the mathematical symbolic language while not guaranteeing that no mathematical notation will appear in the text entry area. LeActiveMath studies on tutoring calculus report on this issue (Callaway et al., 2006; Dzikovska et al., 2006). Structure-rich markup languages, such as MathML or OpenMath,[17] which are typically the internal representation in structured maths notation editors, are too complex to be typed in by dialogue participants. LaTeX, however, combines structured in-line markup and is conceptually simple enough to be suitable for the tutoring setting, especially if the mathematical domain does not involve excessively complex notational constructs. Therefore, while the user interface implemented for the first experiment offered only buttons for entering mathematical symbols, in the second study our interface enabled also LaTeX-like entry of math.

Experimental conditions The main goal of the experiments was to collect data on authentic human–computer tutorial dialogues about mathematical proofs. Thus, with respect to the language behaviour in dialogues in our setting, the experiments were of exploratory nature with the general design facilitating collection of linguistic data. However, both experiments also manipulated one variable related to different aspects of our scenario.

In the first experiment, the exploratory part of was focused on the natural language aspects of the interaction. The experimental part concerned the pedagogical aspects: three tutoring styles – minimal-feedback, didactic, and Socratic – were compared with respect to their effect on learning. In a completely randomised design, subjects were split into three groups and tutored by the same tutor according to three predesigned algorithms. The purpose of the manipulation was two-fold: First, it was to test the effectiveness and completeness of hinting categories formalised for Socratic tutoring before the experiment. Second, it was to identify limitations of the predesigned hinting algorithm and to propose improvements based on data analysis.

[16] FFES (Smithies et al., 2001), Infty (Fujimoto et al., 2003), JMathNotes (Tapia and Rojas, 2004), WebMath (Vuong et al., 2010), and Mathellan (Fujimoto and Watt, 2010) are examples of such interfaces; see also the surveys in (Zhang and Fateman, 2003; Fateman, n.d.a).

[17] `http://www.w3.org/Math`, `http://www.openmath.org`

In the second experiment, we were interested in the factors that might influence language styles in dialogues on proofs. Specifically, we wanted to find out whether students' language production would differ depending on the study material's presentation form. The subjects were split into two groups and, before tutoring, provided with reading material presented in a formal or a verbose style. More details on this aspect of the second experiment follow in Section 2.4.3 of this chapter. The analysis of language production the two conditions will be presented in Section 4.3.2 of Chapter 4.

2.4 Overview of the experiments

The reminder of this chapter summarises the setup of the experiments and presents an overview of the collected data. We first summarise the common aspects, then elaborate on the two experiments, and finally describe the corpora.

2.4.1 Common aspects

In both experiments the subjects were Saarland University students. With the exception of one subject in the second experiment, they were native speakers of German. The non-native speaker had been living in Germany for about 20 years and her German was assessed to be at near-native fluency; data of this subject were included in the analyses. The subjects' prior knowledge in mathematics declared in interviews ranged from little to fair. All the wizards (tutors) were native speakers of German with experience in teaching mathematics.

The subjects were solving proofs with a tutoring system simulated in a Wizard-of-Oz setup described in Section 2.2. During the experiment the subjects and the wizard(s) were seated in separate rooms connected through a voice channel, and with a one-way window between the rooms. In case of technical problems unrelated to solving exercises, the subjects could communicate with an experimenter via a microphone and speakers.

An experiment session started with an introduction to the experiment by the experimenter who informed the subject about the recording and logging setup, explained the procedures, handed out the study material, and demonstrated the interface. The study material was presented on paper and included domain knowledge required to solve the exercises. It was available to the subjects throughout the duration of the session. After the introduction, the subjects filled out a background questionnaire and were allowed a study time.

The proof problems concerned fundamental mathematics. The subjects were not taught a particular proof, but were allowed to propose their own solution. The expectation was that the tutor (wizard) would recognise the subject's line of reasoning and guide the tutorial dialogue accordingly. The subjects were instructed to enter proof steps rather than complete proofs at once in order to prompt dialogue. They were also asked to think aloud while solving the exercises. In both experiments the subjects were audio- and video-recorded.

The subjects were interacting with the simulated system through a GUI which included a designated input entry area for composing messages to the system. The GUI included a button bar with mathematical symbols and a read-only dialogue history area which displayed the previous student and tutor turns. The subjects could enter their utterances using a keyboard (typing) or a mouse (clicking on the mathematical symbol buttons). Before starting a session they were shown the GUI's functionality and allowed a short time to familiarise themselves with the interface.

The subjects were told that they were participating in an evaluation of an intelligent tutoring system with conversational capabilities which could understand and respond in German, thus they could use both natural language and mathematical notation while solving exercises. No restrictions on the form or style of the language were specified during the introduction to the interface. In the *minimal feedback* condition of the first experiment (see ''Tutoring'' in Section 2.4.2) the wizard used precompiled text as responses. In the other tutoring conditions and in the second experiment, the wizard was unconstrained in formulating his turns. After the experiment session, the subjects filled out a survey questionnaire and were informed about the simulation. Participation in the experiments was remunerated.

2.4.2 The first experiment

The setup of the first experiment was the following:

Persons A mathematics graduate with experience in teaching was hired to play the role of the wizard. Before the experiment he was trained on the use of the interface and on the predefined tutoring algorithms. In order to distribute the cognitive load involved in tutoring in the WOz setup, two *helpers*, the authors of the tutoring algorithms, assisted the wizard. The fourth person involved was the *experimenter* who introduced the procedure, answered non-task-related technical questions during the experiment, and debriefed the subjects after the experiment.

Subjects Twenty two subjects participated in the experiment. Their backgrounds were in humanities or sciences. No prerequisites on completed coursework in mathematics were set as criteria for participation. Maths knowledge at the level required for university admission was assumed.

Procedure An experiment session consisted of three phases. First, the subjects were given a pretest. Second, they interacted with the simulated tutoring system. Tutoring was performed in one of the three tutoring conditions described further in this section. Third, the subjects solved a posttest exercise and were debriefed. A three-phase experiment session lasted about two hours.

User interface The graphical user interface developed for the first experiment consisted of three areas: the button bar, the dialogue history, and the input line. The button bar contained buttons with mathematical symbols relevant in the domain. The dialogue history displayed the prior dialogue turns in a non-editable mode. The wizard's interface, aside from the same components, contained a larger main area in which the wizard selected the answer evaluation categories (see ''Tutoring'' further in this section) and hint categories.

Domain and proof exercises The proofs in the first experiment concerned *naïve set theory*. The main reasons for choosing this domain were that, first, naïve set theory is not too complex and so fundamental that not a lot of background knowledge is required and, second, it has been previously formalised for proof automation (Suppes and Sheehan, 1981; Benzmüller and Kohlhase, 1998; Ravaglia et al., 1999a; Benzmüller et al., 2001). For simple problems within its decidable fragment, wrong proof steps can be identified by a model generator by searching for counterexamples (Benzmüller et al., 2001). In this respect naïve set theory is a good domain of choice for a prototype system. The following exercises were used:[18]

Pretest	$K(A) \in P(K(A \cap B))$
Dry-run	$K((A \cup B) \cap (C \cup D)) = (K(A) \cap K(B)) \cup (K(C) \cap K(D))$
Powerset	$A \cap B \in P((A \cup C) \cap (B \cup C))$
Complement	If $A \subseteq K(B)$, then $B \subseteq K(A)$
Posttest	$K(A \cup B) \in P(K(A))$

The **Dry-run**, **Powerset**, and **Complement** proofs were used during the tutoring session. The easy **Dry-run** proof was presented first and served as a warm-up exercise. The remaining two proofs were presented in random order. A time limit of 30 minutes per exercise was imposed.

[18] K stands for set complement and P for powerset.

Study material Subjects were given a handout with mathematical content needed to solve the proof tasks: an introduction to naïve set theory, definitions of concepts, theorems, and lemmata. The was time limit on preparation.

Tutoring The tutoring strategy was the manipulated variable in the first study. The subjects were split into three groups and randomly assigned to one of the three tutoring conditions: *minimal feedback*, *didactic*, and *Socratic*. In the *minimal feedback* condition (control group), the tutor used standardised phrasing to inform the student only of the correctness and completeness of his proof steps. The prescripted phrasing was ''Das ist richtig/nicht richig'' (*This is correct/incorrect*) and ''Das ist unvollständig oder nicht ganz korrekt'' (*This is incomplete or inaccurate*). The tutor did not answer students' questions; the response to all questions was phrased ''Das kann ich nicht beantworten'' (*I cannot answer this*). In the *didactic* condition, the tutor disclosed the next correct step whenever the student would stop making progress or explicitly request help. The tutor answered students' questions. In the *Socratic* condition, the tutor executed a predesigned hinting algorithm to help the student discover the solution by guiding him towards it. The tutor was supported by the helpers, the authors of the Socratic algorithm, in deciding which hint should be realised. Surface realisation of the hints was left to the tutor. The null hypothesis was that the students' performance in the three conditions would not differ statistically. Performance was measured based on scoring the pretest and posttest performance and, unexpectedly, confirmed the hypothesis.[19]

The tutor's responsibilities included the following tasks: (i) evaluating the student's proof step in one of the following answer categories: correct, incomplete accurate, complete partially accurate, incomplete partially accurate, and wrong; the assigned category was saved in the session log file together with the dialogue transcript, (ii) decide what dialogue move to make next (for instance, inform about correctness status, give hints, etc.), and (iii) verbalise it. At the end of each exercise, the tutor summarised the entire proof or, if the student did not complete the proof, presented a valid proof to the student.

2.4.3 The second experiment

Persons Four tutors were invited to play the role of wizards in the experiment; the wizards were effectively also subjects in the experiment: by observing multiple tutors we wanted to find out whether acceptability of different proof

[19] The pedagogical aspects of the experiment have been presented in more detail in (Tsovaltzi et al., 2004; Tsovaltzi, 2010).

step sizes (granularity) varies between teachers. The tutors' background with respect to teaching mathematical proofs was the following:

Tutor 1 Senior lecturer from the Saarland University with several years of experience in lecturing a course *Foundations of Mathematics*

Tutor 2 Professional mathematics teacher, with a few years of teaching experience who participated in our first experiment

Tutor 3 Recent Saarland University graduate with a degree in teaching mathematics

Tutor 4 Doctoral student at the Saarland University Institute of Theoretical Mathematics with several years of experience as a TA in various mathematics courses

One *helper* was operating the audio and the video equipment, overseeing the recording and the technical side of the experiment in general. Two *experimenters* took turns in taking the responsibility for communicating with the subjects. The experimenter also decided which exercises the subject should solve (see ''Domain and proof exercises'' further in this section).

Setting The subjects and the experiment team were seated in separate rooms. The wizards and the experimenter could see the subject on a display transmitting signal from a dome-camera in the subject's room. The subject's computer was running screen capture software. In the original setting the wizards could not see the screen capture feed as we did not want them to be influenced by subjects' false starts which were not submitted to the system. However, already on the first day of the experiment, it turned out that mathematical expressions produced by subjects were so complex that wizards' response times became unacceptably long. Since the wizards knew that short response time was important, under this stress condition there was more chance for mistakes in evaluating subjects' contributions. We therefore decided to transmit the screen capture feed to an additional display, so that the wizards could start evaluating the expressions as the subjects typed. In some cases of extremely long formulas this proved critical in making the wizards' task feasible.[20]

Subjects Thirty seven students with different educational backgrounds participated in the experiment. A prerequisite for participation was to have taken at least one university level mathematics course.

Procedure Before tutoring, the subjects were shown how to operate the interface, presented with the study material, and allowed 25 minutes study time. Next, they interacted with the simulated system. The subjects were then debriefed and filled out a questionnaire. A session lasted about two hours. Pretests and posttests were not administered due to time constraints. Conducting

[20] We will return to the formula length problem in Chapter 4 (Section 4.3.2).

further experiments was unfortunately impossible for logistic reasons. Lack of test data did not allow us to perform an analysis of the relation between the linguistic properties of students' discourse and learning; see, for instance, (Ward and Litman, 2006) for an interesting study on cohesion.

User interface The interaction between the subject and the wizard was mediated by a chat environment built on top of a customised version of TEXmacs, a LATEX editor operating in the *what you see is what you get* mode.[21] The interface offered multiple options for inserting mathematical expressions: LATEX commands (`\cup` for set union, etc.), their German counterparts (`\Vereinigung` for set union, etc.) as well as traditional GUI buttons. The editor supports *copy–paste* functionality which enabled copying text from the prior dialogue. Dialogue history was displayed in *read-only* mode. The available mathematical expression commands were printed on a handout. Before the session, the experimenter instructed the subjects on using the GUI and showed the different formula input modes. The subjects had a few minutes time to familiarise themselves with the GUI. The session log files contain information on the mode in which mathematical expressions were inserted.

Domain and proof exercises The proof exercises were in the domain of binary relations. Theorems and definitions in binary relations build on naïve set theory and the conceptual complexity of the domain is comparable to naïve set theory. The reason for choosing a new domain was, among others, to facilitate testing of the scalability of the input interpretation component.[22]

The subjects were asked to prove the following four theorems:

Let R, S, and T be binary relations on a set M.

Exercise W $(R \circ S)^{-1} = S^{-1} \circ R^{-1}$
Exercise A $(R \cup S) \circ T = (R \circ T) \cup (S \circ T)$
Exercise B $(R \cup S) \circ T = (T^{-1} \circ S^{-1})^{-1} \cup (T^{-1} \circ R^{-1})^{-1}$
Exercise C $(R \cup S) \circ S = (S \circ (S \cup S)^{-1})^{-1}$
Exercise E Assume R is asymmetric. If R is not empty (i.e. $R \neq \emptyset$), then $R \neq R^{-1}$

Exercises **W**, **A**, **B**, and **C** were selected in such way that once solved they may be used as justifications in the subsequent proofs. **C** is a theorem if S is symmetric, but not in the general case. The subjects were expected to provide an argument for this. **W** was a warm-up exercise and **E** was presented only to those subjects who had difficulties completing the initial exercise.

[21] `http://www.texmacs.org` [Accessed: 2006]

[22] Results will be shown in Chapter 7.

The subjects started with exercise **W** and normally followed with **A**, **B**, and **C**, in this order. The experimenter was monitoring progress on a screen capture display. If he noticed that a subject was struggling with the warm-up exercise, he could at any time ask the subject to stop and move on to **E**. Once **W** was completed or the subject was asked to proceed to **E**, he could spend as much time on the exercise(s) as needed. There was no time limit on the completion of individual exercises, however, sessions were kept to about 2 hours.

Study material The content of the study material, adapted from (Bronstein and Semendjajew, 1991), reviewed definitions and basic theorems in binary relations. Inspired by findings on alignment effects observed in human–computer dialogues (see discussion in Section 2.2, p. 63), we wanted to find out whether a similar effect would be induced by the presentation style of the study material in computer-based tutoring. To this end, two study material versions were prepared: in one content was presented in a *formal* way, using mainly formulas, in the other the same content was presented in a *verbose* way avoiding formal notation and using natural language instead. Figure 2.1 (p. 78) illustrates the difference in the presentation of the definition of the subset relation.

The subjects were randomly assigned to the formal (FM group) or verbose (VM) study material condition and given the corresponding handout. They were also provided with an example proof, shown in Figure 2.2 (p. 78), formulated using natural language and formulas, and allowed 25 minutes to revise. Our hypothesis was that the language the subjects would use to solve exercises would reflect the study material's format, that is, the subjects would ''align'' to the presentation format. This hypothesis was confirmed.[23]

Tutoring The second experiment had two objectives: the first was to obtain more linguistic data on proofs and to verify our hypothesis concerning language production. The second objective was to obtain data on pedagogically acceptable granularity of proofs – that is, argumentative complexity: the level of detail in proofs, the number of reasoning gaps which can be left in – in a tutoring setting. To this end, we asked the tutors to indicate explicitly their judgements on granularity of every proof step the students proposed. By analysing tutors' granularity judgements, we wanted to find out what characterises pedagogically acceptable and unacceptable steps, whether acceptability differs between tutors, and how the granularity compares with the level of detail required by automated deduction systems, specifically, the Ωmega system (Siekmann et al., 2003).[24]

[23] The analysis of the language production in the two conditions is discussed in Chapter 4.

[24] For a data-driven model of proof step granularity based on our corpus see (Schiller et al., 2008).

Sind A, B Mengen und gilt $\forall x(x \in A \Rightarrow x \in B)$, so heißt A eine *Teilmenge* von B. Man schreibt dafür $A \subseteq B$.

(If A and B are sets and $\forall x(x \in A \Rightarrow x \in B)$ holds, then A is called a subset of B. We write $A \subseteq B$.)

Sind A, B Mengen und gilt daß jedes Element von A auch Element von B ist, so heißt A eine *Teilmenge* von B. Man schreibt dafür $A \subseteq B$.

(If A and B are sets and every element of A is also an element of B, then A is called a subset of B. We write $A \subseteq B$.)

Figure 2.1: Definition of the subset relation in the formal (left) and verbose (right) presentation in the second experiment.

Theorem
Sei R eine Relation in einer Menge M. Es gilt: $R = (R^{-1})^{-1}$

Beweis
Eine Relation ist definiert als eine Menge von Paaren. Die obige Gleichheit ist demnach eine Gleichung zwischen zwei Mengen. Mengengleichungen kann man nach dem Prinzip der Extensionalitaet dadurch beweisen, dass man zeigt, das jedes Element der ersten Menge auch Element der zweiten Menge ist. Sei also (a, b) ein Paar in $M \times M$, dann ist zu zeigen $(a, b) \in R$ genau dann wenn $(a, b) \in (R^{-1})^{-1}$. $(a, b) \in (R^{-1})^{-1}$ gilt nach Definition der Umkehrrelation genau dann wenn $(b, a) \in R^{-1}$ und dies gilt nach erneuter Definition der Umkehrrelation genau dann wenn $(a, b) \in R$, was zu zeigen war.

(Let R be a relation on a set M. It holds that $R = (R^{-1})^{-1}$ A relation is defined as a set of pairs. The equation above expresses an equality between sets. Set equality can be proven by The Principle of Extensionality. We show that every element of one set is also an element of the other set. Let (a, b) be a pair in $M \times M$. We have to show that $(a, b) \in R$ if and only if $(a, b) \in (R^{-1})^{-1}$. $(a, b) \in (R^{-1})^{-1}$ holds by definition of the inverse relation if and only if $(b, a) \in R^{-1}$. This in turn holds by the definition of the inverse relation if and only if $(a, b) \in R$, which was to be proven.)

Figure 2.2: Example proof from the second experiment

Table 2.1: Number of subjects per tutor and study material condition in the second experiment

Tutor	No. of subjects FM-group	 VM-group	Row totals
Tutor 1	2	4	6
Tutor 2	8	2	10
Tutor 3	6	6	12
Tutor 4	4	5	9
Column totals	20	17	37

The tutors were presented with general guidelines on Socratic tutoring, but unlike in the first experiment, they were not provided with any tutoring algorithm. There were no restrictions on the tutors' language production. The tutors were asked to annotate the students' proof contributions with answer categories along three dimensions: correctness (correct/partially correct/incorrect), relevance (relevant/limited relevance/not relevant), and granularity (appropriate/too detailed/too coarse-grained). Annotations were inserted during the tutoring session, however, they were not visible on the subject's end of the interface. The tutors were also provided with a headset microphone and asked to record a spoken commentary on their responses. This gave us a record of justifications of tutors' decisions and their comments on the tutoring process.

Table 2.1 shows the number of subjects per tutor and study material condition. The assignment of study material format to subjects and of tutors to subjects was quasi-random; the tutors did not know to which experimental condition a given subject was assigned.[25]

2.5 Overview of the corpora

The main output of the experiments are two corpora of human–computer tutorial dialogues on mathematical proofs. The first corpus, C-I, comprises 22 dialogue log files. Aside from the students' and tutors' turns the log files include time-stamps for each turn, answer category annotations for student turns, and hint category annotations for tutor turns. There are 775 turns in total, of which 332 are student turns (43%) with 443 utterances.[26] The second corpus, C-II, comprises 37 log files with time-stamp information, answer category annotations, and the information on the mode in which mathematical

[25] Distribution of subjects between study material and tutors is not uniform due to subject dropout and due to an error in the WOz software in the beginning of the experiment.

[26] The criteria for utterance-boundary annotation will be presented in Chapter 4 (Section 4.2.1).

Table 2.2: Basic descriptive information on the two corpora

	C-I (Set theory)	C-II (Binary relations)
Subjects/Sessions	22	37
No. Turns	775	1906
Mean No. turns per session (*SD*)	35 (12)	51 (19)
No. students' turns (% No. turns)	332 (43%)	927 (49%)
Mean No. students' turns per session (*SD*)	15 (6)	25 (10)

symbols were inserted. C-II consists of 1906 turns of which 927 are student turns (49%) with 1118 utterances. Table 2.2 summarises basic descriptive information on the two corpora. Figures 2.3 and 2.4 at the end of this chapter (pp. 82 and 83) show example dialogues from C-I and C-II, respectively. In the figures and throughout this thesis, where relevant, student and tutor turns are labelled ''S*m*'' and ''T*n*''; m and n denote turn numbers. If it is clear from the context that students' language is meant, ''S'' labels are omitted.

2.6 Summary and conclusions

We presented two experiments conducted with the goal of collecting data on authentic human–computer tutoring of mathematical proofs. In order to motivate the experiments, we first discussed experts' and learners' proofs and pointed out differences between them. We outlined alternative sources of data in dialogue research and motivated the decision to conduct data collection experiments, rather than to refer to existing sources, such as textbooks or available tutoring corpora. We also discussed the differences between human–human and human–computer interactions which justified the decision for the human–computer, rather than the human–human setup of the experiment. We presented a general overview of the simulation methodology we pursued and motivated our experimental design decisions.

The key lesson learnt from the experiments is that mathematics is a difficult domain for the Wizard-of-Oz setup. First, mathematical proofs are demanding on the wizard. Given that the response time is of major importance in a simulation, the wizard needs support in reconstructing the students' reasoning. In the first experiment, helpers were assisting in making sure that the students' utterances are correctly checked. In the second experiment, we found out early on that the tutors had difficulties visually parsing long mathematical expressions produced by the learners and consequently responses were delayed. Some of the wizards voiced this issue themselves. Therefore, we changed our original setup in the course of the experiment to allow the wizards to see the subjects' input as they typed by transmitting the screen capture output in real-time to an additional computer monitor in the wizards' room. It is interesting that this cognitive overload in processing formulas was observed already in a relatively simple domain. Certainly, one of the main problems was that the mathematical expressions which the students produced were indeed of considerable length. Even simple formula preprocessing, such as syntactic validation, would be helpful here. Perhaps in more complex domains, it would even make sense to let the wizard listen on the subjects' self-talk through an audio channel.

The second observation concerns the user interface. The TEXmacs interface, while certainly more flexible and more convenient for the users than the simple GUI from the first experiment, may have been the ''culprit'' that introduced the problem discussed above. The *copy–paste* mechanism turns out to be a mixed blessing: on the one hand, it is certainly convenient for the users, but on the other hand, copying ad libitum from the dialogue history makes the input not only more prone to errors of sloppiness, but also unnecessarily complex. It is not clear how to cope with this problem. Since *copy–paste* is currently standard in computer programs, suppressing it in a tutoring system appears unnatural. An interim solution could be, for instance, to highlight in some way the copy–pasted parts of the input for the wizard. It is interesting that even though the second interface was better suited for combining natural language and mathematical expressions (formulas could be composed without switching between the keyboard and the mouse) C-I also contains utterances in which natural language and formulas are tightly interleaved.

Our decision as to the mode of interaction, typewritten vs. spoken, was probably right. Technical problems which we experienced during the experiments were easier to solve in this simpler setup. Also, we have two corpora acquired in similar conditions (comparable), so the language processing strategy we propose can be evaluated on both sets of data. As we will show in Chapter 4 the subjects' language production strongly differs between the two data sets; we believe that even the combined set of dialogues is not representative enough for building an input understanding component of a deployable system at this point. While it would be certainly interesting to see how speech and writing could be built into a WOz experiment on tutoring proofs, this would be a project in itself, outside of the scope of this work.

The next chapter, Chapter 3, presents the language of mathematical proofs and reviews the language phenomena which we identified in the dialogues at a qualitative level. The following chapter, Chapter 4, presents a quantitative analysis of the diversity of the students' verbalisations and the differences in the language production in the two conditions in the second experiment.

T1: Bitte zeigen Sie: $A \cap B \in P((A \cup C) \cap (B \cup C))$!
Please show $A \cap B \in P((A \cup C) \cap (B \cup C))$!

S1: $(A \cup C) \cap (B \cup C)) = C$

T2: Das ist falsch. Nach der Distributivität gilt: $(A \cup C) \cap (B \cup C) = (A \cap B) \cup C$. Somit gilt auch: $P((A \cup C) \cap (B \cup C)) = P((A \cap B) \cup C)$. Wie geht es nun weiter?
This is wrong. By distributivity we have: $(A \cup C) \cap (B \cup C) = (A \cap B) \cup C$. Therefore, we also have $P((A \cup C) \cap (B \cup C)) = P((A \cap B) \cup C)$. How does it go on?

S2: $(A \cap B)$ muss in $P((A \cup C) \cap (B \cup C))$ sein, da $(A \cap B) \in (A \cap B) \cup C$
$(A \cap B)$ must be in $P((A \cup C) \cap (B \cup C))$ since $(A \cap B) \in (A \cap B) \cup C$

T3: Was meinen Sie mit: $(A \cap B)$ muss in $P((A \cup C) \cap (B \cup C))$ sein?
What do you mean by: $(A \cap B)$ must be in $P((A \cup C) \cap (B \cup C))$?

S3: da $P((A \cup C) \cap (B \cup C))$ nach Distributivgesetz gleich $P((A \cap B) \cup C)$ ist muss $(A \cap B)$ in $P((A \cup C) \cap (B \cup C))$ enthalten sein.
since by distributivity $P((A \cup C) \cap (B \cup C))$ equals $P((A \cap B) \cup C)$ $(A \cap B)$ must be contained in $P((A \cup C) \cap (B \cup C))$

T4: Was meinen Sie mit: muss $(A \cap B)$ in $P((A \cup C) \cap (B \cup C))$ enthalten sein. Meinen Sie Element oder Teilmenge?
What do you mean by: $(A \cap B)$ must be contained in $P((A \cup C) \cap (B \cup C))$. Do you mean element or subset?

S4: $(A \cap B) \in P((A \cap B) \cup C)$

T5: Meinten Sie vorhin wirklich: $(A \cap B) \in (A \cap B) \cup C$?
Do you really mean $(A \cap B) \in (A \cap B) \cup C$?

S5: ja
yes

T6: Das ist falsch, denn $(A \cap B) \subseteq (A \cap B) \cup C$. $(A \cap B)$ und $(A \cap B) \cup C$ sind Mengen und keine Elemente voneinander. Was folgt als nächstes?
This is wrong because $(A \cap B) \subseteq (A \cap B) \cup C$. $(A \cap B)$ and $(A \cap B) \cup C$ are sets and not elements of each other. What follows next?

...

Figure 2.3: Example dialogue from the set theory corpus (C-I).

...

S9: $(R \cup S) \circ T = \{(a,b)|aRb \wedge aSb\} \circ T$

T9: Das ist nicht richtig!
That is not correct

S10: $= (R \cup S) \circ T = \{(a,b)|aRb \vee aSb\} \circ T$

T10: Die Gleichheit zum vorigen Schritt ist nicht gegeben!
The equality with the previous step does not hold

...

S12: $= \{(a,b)|\exists c(c \in M \wedge ((a,c) \in \{(a,b)|aRb\} \wedge (c,b) \in T)\} \cup \{(a,b)|\exists c(c \in M \wedge ((a,c) \in \{(a,b)|aSb\} \wedge (c,b) \in T)\}$

T12: Ja, das ist richtig!
Yes, that's correct

S13: Wollen zeigen $(S \circ T) = (T^{-1} \circ S^{-1})^{-1}$
Want to show...

T13: Ja. Sie koennen jedoch Ergebnisse von vorhergehenden Aufgaben benutzen.
Yes. Note that you can use the results of the previous exercises.

...

S17: Damit gilt: $(T^{-1} \circ S^{-1})^{-1} \cup (T^{-1} \circ R^{-1})^{-1} = (S \circ T) \cup (R \circ T) =$
With that it holds...

T17: Korrekt!
Correct!

S18: $= (R \circ T) \cup (S \circ T)$

T18: Auch korrekt!
Also correct!

S19: Nach Aufgabe A ist dies $(R \cup S) \circ T$
By exercise A this is...

T19: Genau richtig!
Exactly correct!

S20: Damit folgt die Behauptung
With that the theorem follows

T20: Gut! Vielen Dank! Druecken Sie nun den Knopf fuer die naechste Aufgabe!
Good! Thank you! Press the button for the next exercise!

...

Figure 2.4: Example dialogue from the binary relations corpus (C-II).

Chapter 3

Language phenomena in mathematical proofs

In this chapter we discuss language phenomena in students' proofs. The discussion is based on an analysis of the corpora presented in Chapter 2, however, where relevant, we point out that certain phenomena occur systematically both in mathematical prose and tutorial dialogue. We show that the range of linguistic phenomena in dialogues includes those found in textbooks, but also a range of phenomena specific to the dialogue setting. Language phenomena are classified with respect to their lexical, syntactic, semantic, and context-dependent nature, and exemplified with utterances from the corpora.

The presentation of language phenomena is preceded by an introduction in which mathematical language is presented from two perspectives: as a *special language* and as a language acquired in parallel with mathematical understanding. We characterise the properties of special languages, so-called sublanguages, to show that the language of mathematics can be considered a sublanguage, that certain phenomena we identify in our data are its features as a member of this class, and that therefore they are likely to be found in other corpora of mathematical discourse as well.

Next, we refer to observations from cognitive science of mathematics in order to point at a relation between the language used to communicate mathematics and the stage of mathematical understanding. The model proposed by Tall, which we summarise, suggests that certain phenomena in the students' mathematical language – specifically, *imprecision of linguistic expression* leading to *ambiguity* – may recur *because* they are linked to the level of understanding. Again, this lets us conclude that certain linguistic phenomena in students' language have a systematic nature and prioritise modelling those phenomena in a discourse processing architecture.[1]

[1] The language of mathematics has been subject of analysis, motivated by goals similar to ours in the doctoral dissertations of Zinn (2004), Natho (2005), and Ganesalingam (2009). We will sometimes refer to those works in order to avoid repetition, however, certain overlap is

3.1 Introduction

In the following two sections, we briefly present mathematical language from two perspectives: as a sublanguage and as a language acquired in parallel with mathematical understanding. These two views help *explain* some of the phenomena observed in the corpora.

3.1.1 Mathematical language as a special language

Language is a type of code. Natural language is a code which enables communication of meanings by means of words. From the perspective of its purpose as a means of communication, language is a system consisting of a vocabulary and grammar rules that makes linguistic behaviour possible. A *sublanguage*, or *special language*, as opposed to the *general language*, is a language used by a particular community (social or professional, for instance) or used to talk about specialised topics, a limited subject matter, for example, within a particular discipline (Harris, 1968; Sager, 1972; Hirschman and Sager, 1982; Grishman and Kittredge, 1986).

Sublanguages tend to diverge from the general language in that they are characterised by a systematic recurrence of non-standard or even ungrammatical structures, stylistic patterns, high frequency of certain constructions, conventionalised phrasings, by the use of specially created terminological systems and special written notation whose verbalisation may require adhering to commonly agreed rules (Kittredge and Lehrberger, 1982; Linebarger et al., 1988; Grishman and Kittredge, 1986). Typical examples of special languages are the language of law, with its characteristic style and choice of wording, hardly comprehensible to the layman, the language of medicine and pharmacology, with their Latin terminology and frequent use of abbreviations, or the language of chemistry. The latter is particularly interesting in that it has developed different code systems to refer to chemical elements and compounds, the first-class entities in the world of chemistry; for example, referring to the substance commonly known as *water* we can say or write *hydrogen monoxide* using a technical term (linguistic code), or H_2O (symbolic code), or draw a graphical representation of the compound's structure (visual code). The formal language of mathematics can be considered a special language which, much like the language of chemistry, combines a subset of a natural language with a special kind of written code whose vocabulary, unlike that of natural language, does not consist of words (in the sense of words of English

unavoidable. The discussion of language phenomena presented in this chapter benefited from monographs and articles on mathematical discourse by Halmos (1970), Steenrod et al. (1973), Knuth et al. (1989), and Bagchi and Wells (Bagchi and Wells, 1998; Wells, 2003, n.d.).

or German), but solely of special symbols typically limited to numbers, letters, multi-character abbreviations, and graphical signs, which can be combined according to prescribed rules to form expressions of arbitrary complexity. This written symbolic code is a kind of conventionalised notational system that makes *rigorous* and *formal* mathematics possible.

The mathematical language we know from school classes, university lectures, and textbooks – the *informal* mathematical language – certainly does not consist of the symbolic language alone. Especially while teaching and learning we do not use such a linguistically limited form of expression to communicate mathematics. In fact, the symbolic notation often constitutes a serious cognitive barrier in understanding mathematical concepts (Moore, 1994; Dorier et al., 2000; Booker, 2002; Downs and Mamona-Downs, 2005). The language we do use, ever since we first encounter mathematics in preschool, is our mother-tongue. We start by informally talking about mathematical objects in natural language in order to understand the concepts intuitively. Gradually, we learn the mathematical terminology – the technical terms that name the concepts – observe that certain common words from everyday vocabulary name mathematical notions, acquiring ''mathematical meaning'', and we adopt the new usage. At the same time, much like learning a foreign language, we learn the new language of mathematical notation and combine it with natural language. This process of learning the ''mathematical language'' is not a trivial one, but the success in understanding mathematics has been shown to crucially depend, among others, on the learner's ability to master the ways of mathematical communication; Sfard (2000, 2001) views the process of learning mathematics as developing a special type of discourse.

Efficient communication of mathematics relies heavily on the interaction of the two languages: the natural language (linguistic code) and the language of mathematical notation (symbolic code). The two languages can be thought of as two *modes* of expression which can be not only flexibly exchanged, but also interleaved. In this sense, informal mathematical language can be considered ''multi-modal''; the symbolic and natural language modes are integrated into the syntax of the special language of mathematical discourse.

It is useful to realise in the context of mathematics tutoring that mathematical style and language, in particular, the level of formality in expressing mathematical statements, *evolves* as learners develop deeper mathematical understanding. Tall (2004b) refers to the different stages of mathematical cognitive development as *three worlds of mathematics* and explicitly points at a relation between the stage of understanding in the course of learning and the properties of the language used to communicate mathematics. In the next section we briefly review Tall's theory.[2]

3.1.2 Learning mathematics and mathematical language

From the point of view of cognitive development, understanding (also mathematical) and creative thinking is crucially dependent on three basic human cognitive activities: *perception*, *action*, and *reflection* (Skemp, 1987, 1979). Perception is concerned with *objects* and their *attributes*. Objects can be manipulated using acquired *action schemas* which, in turn, can themselves be perceived as objects (in the sense that they are mental units) and become subject of thought processes. More sophisticated mental objects can be formed through *reflection on perception and actions*. This stepwise development model is based on the Piagetian tripartite theory of abstraction: empirical (objects), pseudo-empirical (actions), and reflective (actions and operations as objects of thought) (Piaget, 1985). Other stratified models are conceptually related in that they share the underlying common distinction between the three stages of cognitive development: interaction with the environment (enactive stage), mental representations and operations on them (iconic thinking), and abstract reasoning (symbolic/formal thinking). In mathematics, for instance, the notion of a number, the construction of natural numbers, and the extension of the notion of a number (cardinal numbers) are based on abstraction and generalisation using sets (objects) and counting (action) and form an axiomatic and definitional basis for formal proofs in domains in which numbers are objects.

Building on existing established theories of cognitive development, David Tall formulated a theory of mathematical thinking in terms of (not necessarily sequential) transitions between *three ‘‘worlds'' of mathematics* which are distinct, but interrelated, and which reflect the tripartite structure of cognitive development outlined above. He claims that the three ‘‘worlds'' are characterised by different mechanics and ways of operating, different forms of proof, and, most interestingly in our context, *different use of language*.

Tall's Three Worlds of Mathematics[3] The conceptual–embodied or *embodied world* is the world of experiences with our physical and mental reality: our perceptions of things we sense and interpret. Early conception of

[2] Incidentally the structural ambiguity in the reading of ‘‘and'' in the next section's title is actually appropriate: on the one hand, the theory points at a dependency between *learning mathematics* and the *mathematical language* used at different stages of learning, on the other hand, it is concerned both with *learning mathematics* and with *learning mathematical language*.

[3] The following two paragraphs summarise the main ideas from (Tall, 2004a) and (Tall, 2004b).

numbers and arithmetics are largely set in the embodied world: a single object is associated with the number one, a group consisting of one object and another object, with the number two, etc. Early counting is also embodied. Through reflection and development of language, we can envisage idealised concepts which do not exist in reality, for instance, an infinite line that is perfectly straight and infinitely thin or non-euclidean geometries.

The second world, proceptual–symbolic or *proceptual world*, is the world of symbols used for calculations. Their crucial property is their dual role: that of denoting *processes or actions* and *concepts*. For instance, the notation $1 + 1$ represents both the process of addition (counting) and the concept of a sum (an action encapsulated in a concept representing the result of counting). This dual nature of mathematical symbolism has been also emphasised by Sfard (1991). Within the proceptual world we move to more involved number concepts: from fractions and negative numbers through rational and irrational numbers to complex numbers. Complex numbers and operations on them are examples of evidence that symbol manipulation can be performed without reference to the embodied world. They can be, however, also represented as points in a plane, giving them an embodied interpretation. An abstraction of the notion of mathematical operation leads also to more sophisticated general concepts, such as limits.

The third world, the formal–axiomatic or *formal world*, is the world of formal definitions that specify properties of mathematical structures (for instance, groups, fields, vector and topological spaces) using formalised axioms. There are no embodied representations in this world, only formal symbolic representations. New objects can be defined using existing axiomatic definitions and their properties can be deduced in formal proofs through which new theorems can be established, thus building new coherent formal theories.

The embodied world, inhabited by objects and actions on them, is thus linked to the basic activity of perception. The proceptual world with actions on objects, reflections on these actions and their symbolic representations (which, in turn, are also objects that can be processed) is linked to the basic activity of performing an action. Finally, the formal world of axioms can be linked to the activity of reflection upon the properties and relationships between the objects in the embodied and the proceptual worlds. Tall points not only at the fact that the three worlds reflect the different ways of understanding mathematics, but also at the fact that language operates differently in each of these worlds.

The language in the Three Worlds of Mathematics In the embodied world the use of language starts with references to everyday experiences with mathematical objects. Once basic categories of *objects are named* (''point'', ''line'', ''circle'', ''square'', or ''triangle'') their *properties are described*: for instance, squares and triangles ''have sides''; squares are ''four-sided figures with all sides equal and (at least) one right angle'', and so on. Moreover, similar or related objects can be prescribed: a ''four-sided figure with opposite sides equal and (at least) one right angle'' is a ''rectangle''. With such descriptive definitions focusing on properties of objects a learner can build first complex object hierarchies; squares are special kind of rectangles, for instance. In the embodied world *the language is mainly used as a descriptive and prescriptive tool*. The linguistic devices include (complex) noun phrases that name concepts, property-naming adjectives, adverbs that further qualify properties, and basic common verbs (such as ''is'', ''has'', ''contains'') to talk about relations between the objects.

The action-based proceptual world needs language which can *talk about actions* (processes or algorithms, for instance) and which includes derived or related lexical forms to *talk about objects that correspond to the actions*. For the process of counting we need ordinal and cardinal numbers, for summation or adding, we need the notion of a sum, etc. The conscious use of the flexibility of language to name processes and concepts represented symbolically and the realisation that symbols denote both processes and concepts is a major factor in mathematical comprehension, in particular, in developing calculating and symbol manipulation skills. An additional function of language in the procept world is *to narrate* or *report on the conducted operations* (for instance, in the form of a self-talk or an internal monologue), *to specify operations that need to be performed*, and *to manage progress* (by asking questions, stating completion of calculations, etc.) The main function of the processes is to perform calculations, while *the main function of the language is to perform speech acts that correspond to the calculations*; hence the use of ''action'' verbs, performative speech acts, and the imperative mood in the internal monologue.

The formal world uses *technical language*. It is based on everyday language, however, if everyday words are used, they are used in a precisely defined technical sense: a *field* is not a kind of area, the word ''set'' is not synonymous with ''group'', an *identity* does not care about its psychological identification, ''group theory'' is not another name for the theory of the crowd, and a *zero ideal* is not an oxymoron. Aside from common words with new technical meaning, the formal language uses *technical terminology* invented specifically for the given mathematical domain or reserved for technical use; in the ''real world'', it would sound rather odd to remark casually about a woman: ‘‘I like her deep brown eyes and the gentle *ellipsoid* of her face.’’ Finally, the formal world, is the world of *symbolic language*. Definitions, theorems, and proofs in the formal world refer to axioms unambiguously expressed in a formal notation. Here, *the language is a means of formalisation.* A peculiar characteristic of the formal world is that the structures defined in terms of axiomatic properties do not at all need to have corresponding embodied counterparts.

The point of this somewhat lengthy introduction to the chapter was to show that because of the nature of mathematical language as a special language and given the type of user we have in mind, a mathematics *learner*, a lot of the phenomena we will describe can be considered universally characteristic of our setting. Tall’s theory, in particular his observations on the students’ language, explain some of the phenomena in our mathematical dialogues: the use of imprecise language to express mathematical concepts (discussed in Section 3.2.2.4), the use of certain types of anaphora in referring to objects expressed in symbolic language (Section 3.2.2.5), verbalisation of symbolic expressions (Section 3.2.1.2), or the action verbs ‘‘narrating’’ proof construction (Section 3.2.2.4). Moreover, and most importantly, they point at the fact that these properties of the language (its imprecision, recurrence of certain reference phenomena, the occurrence of action verbs) are an inherent part of (students’) verbal expression in mathematics. Thus, the phenomena we discuss in the next section, in particular, those characteristic of informal language, are not specific to our corpora alone, but rather can be expected to be found in other corpora of students’ mathematical language as well.

3.2 The language of mathematical proofs

Natural language can be considered inherently unsuitable for mathematics because its interpretation is strongly dependent on context and because of its notorious main flaws – imprecision and vagueness – which tend to lead to ambiguities in interpretation. Yet, in spite of these ‘‘imperfections’’, natural language was for a long time the sole medium for communicating mathematics.

Before symbolism was introduced in the sixteenth century, all of mathematics was done in ordinary language. In early algebra, solutions to what we now know as polynomial equations were presented as worded rules in Arabic.

In his *Short book of al-jabr and al-muqābala*, al-Khwārizmī, an eighth century Persian mathematician, considered quadratic equations such as this:

Property and ten things equals thirty-nine

($x^2 + 10x - 39 = 0$ in today's notation) and presented solutions as follows:

> *Take the half of the number of things, that is five, and multiply it by itself, you obtain twenty-five. Add this to thirty-nine, you get sixty-four. Take the square root, or eight, and subtract from it one half of the number of things, which is five. The result, three, is the thing.* (Kvasz, 2006, p. 292)

In the sixteenth century, Cardano still worked with worded equations (*cubus and thing equal number* for $x^3 + bx - c = 0$; *ibid.*) and it was not until Descartes and Viète that the first symbolic language for equations and manipulation of formulas was introduced. However, counting, numbers, simple calculations, and ‘‘natural language mathematics’’ had existed since the Babylonian civilisation (ca. 2000–1600 BC); even earlier, since the Sumerian times (ca. 3000–2300 BC) already. Al-Khwārizmī's description of finding the unknown is in fact perfectly comprehensible even if it sounds more like a worded recipe or an algorithm[4] (for a method known as ‘‘completing the squares’’) rather than the kind of solution with which we are more familiar nowadays (using the discriminant).

What the example illustrates is that natural language, however imprecise, is flexible and remarkably expressive in that using words (nouns, indefinite and definite descriptions, cardinals) we can *name* (abstract) objects and we can further *refer* to these objects in the subsequent discourse using a range of linguistic devices. For instance, in the English translation of the reproduced text, the noun phrase ‘‘the half of the number of things’’ introduces a new entity of a number type into the discourse as well as refers to an entity previously introduced with the noun phrase ‘‘ten things’’ in the problem description. The new entity is further referred to with its name, ‘‘five’’, in the parenthetical clause and evoked again with a pronoun ‘‘it’’. In order to follow the solution, the reader must just keep track of the discourse referents, much like in ordinary discourse, and perform the mathematical operations simultaneously. Natural language words such as ‘‘a thing’’, ‘‘something’’ serve as placeholders, or natural language *variables*, for which no symbolic representation existed at the time. The introduction of symbolism for variables by Viète lead to a revolution not only in written mathematics, but also in mathematical thinking.

[4] Nota bene, the origin of the word is al-Khwārizmī's name.

Unlike natural language, the symbolic language of mathematics has not been evolving over many centuries. Most of basic algebra and calculus notation was established in the seventeenth and eighteenth centuries by Oughtred, Leibniz, and Euler and conventionalised to a large extent within a short time. Set theory notation is due to Peano and Cantor (late nineteenth century) and Russell, Landau and Bourbaki (twentieth century). Most of the calculus notation is due to Leibniz and Euler (late seventeenth and eighteenth century), and to Gauss, Weierstrass, and Cauchy (from the nineteenth century on).[5]

In the following sections we ‘‘deconstruct’’ the language of mathematics. The analysis is performed from point of view of a computational linguist whose aim is to design and implement a language processing architecture for mathematical discourse. The task of the interpretation component in such an architecture is to bridge the gap between the informal language of proofs and a formal language of a mathematics assistance system which performs reasoning tasks (a proof checker or an automated theorem prover); see Section 1.2. Considering these practical aims, philosophical aspects of mathematics and mathematical discourse – the nature of the universe of discourse, the existence of mathematical entities – will not be even touched upon here.

We first analyse the symbolic component alone (Section 3.2.1) and then the familiar informal mode in which natural language is interspersed with symbolic notation (Section 3.2.2). The sections have a similar structure: we break the language down to its lexicon, its syntax, semantics, and discourse-pragmatic, context-related phenomena. Most of the example utterances are directly quoted from our corpora, preserving the original spelling and capitalisation; some of the quoted mathematical statements are also false. In the English translations we attempt to reproduce the phenomena present in the German originals in order to show that they appear across languages, however, where this is difficult or impossible, we provide additional explanation.

3.2.1 The symbolic language

According to the oft-repeated slogan, all mathematics is is a language. On a cursory look, in a mathematical paper or textbook one sees hardly anything but its ‘‘alien’’ symbol system which typically stands out displayed in indented formulas centred on the page. The title of Ervynck’s detailed analysis of mathematical symbolic language and its syntactic structure, *Mathematics as a foreign language*, emphasises precisely this point (Ervynck, 1992). In this

[5] Cajori’s *A History of Mathematical Notations* is *the* classic source on the subject of mathematical symbolic language. A resource on the earliest uses of mathematical symbols is maintained at `http://jeff560.tripod.com/mathsym.html` [Accessed: 2007].

section, we analyse the symbolic language of mathematics from a linguistic point of view: we look at its lexicon, syntax, discuss semantic and pragmatic phenomena, in particular, its ambiguity, surprising imprecision, context- and convention-dependence, and ''ungrammaticality'' (ill-formedness) in symbolic expressions constructed by learners.

3.2.1.1 Lexicon

The mathematical symbols' vocabulary typically includes the lowercase and the uppercase (stylised) letters of Latin, Greek, and exceptionally old German and Hebrew alphabets, numbers, multi-character abbreviations, and a range of non-alphanumeric iconic signs and punctuation symbols. Unlike in natural language, arbitrary identifiers can be defined to stand for any concept so long as consistency is maintained. Of course, arbitrary reassignment of known symbols or assignment of new symbols to concepts for which exiting symbols are widely used would be counter-productive and might introduce unnecessary confusion, therefore it is not practised.

Letters, numbers, and their bracketed sequences name mathematical ''individuals'' in a given domain (be it primitive objects or complex structures, such as (x, y) or $\{1, 2\}$) and constitute *atomic terms* of the formal language. In principle, the symbolic vocabulary is infinite: letters can be subscripted or superscripted with numbers or punctuation (typically apostrophes) to obtain an infinite repository: x, x_1, x_2,... or x', x'', x''',... In practice only a small subset of the infinite lexicon is mentioned explicitly; infinite collections of objects are marked with an ellipsis symbol (like in the preceding sentence).

Mathematical operators (relation, function, and binder and quantifier symbols) are typically represented by iconic signs ($=$, $\sqrt{}$, $<$, $\subset$, $+$, $\cup$, $\vee$, $\forall$, etc.), accent- and punctuation-like symbols ($\widehat{}$, $\prime$, !), mnemonic abbreviations ($\lim$, $\sin$, $\mathtt{Im}$) and letters (Σ, Π, ∂, d). New abbreviations and graphical signs are continuously introduced as new mathematical objects are being defined.

Operators come with the notion of *arity*, that is, information on the number of arguments they take, and with information on the types of operands to which they can be applied; this is analogous to predicate–argument structures of natural language relational lexemes and sortal restrictions on their arguments. In standard mathematical texts, the addition operator, $+$, for example, takes exactly two arguments, while the summation operator, Σ, three arguments: the conditions on the lower and upper summation bounds and the expression representing the terms being added, of which the first two (the summation bounds) can be left implicit if they are clear from the context; this is often the case if summation ranges from minus to plus infinity, for instance, or if the summation range is given in the text preceding the occurrence of the symbolic expression.[6] The sortal restrictions on the operands are specified by the domain of the concept (relation or function) for which the operator stands in the given context. The domain, in turn, is specified in the concept's definition. The previously mentioned $+$-symbol, for instance, is typically defined as an addition operator in (all) number domains, hence, the expression $\pi + e$ does not violate the sortal restrictions if by π and e we mean the two real numbers, however, the corresponding operation on sets is denoted by the set union operator, $\cup$.

Much like natural language needs punctuation symbols, the comma and the full-stop, to delimit clauses and sentences, mathematical language uses parentheses and brackets (square, curly, angle brackets) to delimit the scope of mathematical operators. In some formal texts, a square or a bolded dot is used as an additional scope-defining punctuation in order to reduce the number of parentheses.[7] Brackets have also a grouping function in the notation of mathematical objects. For instance, by convention, pairs are enclosed in round parentheses, while sets in curly brackets ($(1,2)$ is an ordered pair with 1 as the first and 2 as the second coordinate, while $\{1,2\}$ is a set with these elements).

Also certain punctuation-like symbols serve to denote mathematical concepts. For instance, single vertical lines denote the absolute value of an expression ($|x|$) and pairs of vertical lines, the norm of a vector ($\|\mathbf{x}\|$). Primes and accents (circumflex, check, tilde) tend to have a modifying function: they introduce an object in some way related to the object they modify. Likewise, functionally related objects often receive the same letter names distinguished by primes or accents; for instance, in f', a prime marks the derivative of a function f, $\hat{X}$ might be chosen to name the closure of X. Primes also mark collections of objects of the same type: x', x'',...

Horizontal and diagonal lines may also act as typographical separators, as in the set comprehension notation ($\{x|x > 7\}$) or in the notation for fractions ($\frac{7}{17}$ or $7/17$). The comma is used in enumerations, much like in natural language: $\forall x, y \neq 0$...

3.2.1.2 Syntax

Mathematical expressions are built according to rules of syntax which are often introduced only informally. In mathematics textbooks, examples of expressions with particular operators are typically presented together with the definition of

[6] We will return to the role of context in Section 3.2.1.4.

[7] Saving on parentheses is common in logic and meta-mathematics; see, for example, the use of dots in *Principia Mathematica*.

the given concept and with natural language phrases illustrating how the given expression is to be ‘‘pronounced’’, as in the following definition from Bartle and Sherbert (1982):[8]

> *If A denotes a set and if x is an element, we shall write*
>
> $$x \in A$$
>
> *as an abbreviation for the statement that x is an* ***element*** *of A, or that x is a* ***member*** *of A, or that x* ***belongs*** *to A, or that the set A* ***contains*** *the element x, or that x* ***is in*** *A.*

In formalised systems, such as formal logic or proof theory, the syntax of the formal language (the complete range of licensed expressions, or *well-formed formulas*) is explicitly introduced inductively. Inductive syntax definitions follow a definition schema that starts with an introduction of atomic terms (constants and variable symbols and conventions for obtaining an infinite set of those; for instance, using primes or numerical subscripts), followed by a definition of complex terms (including operator symbols that combine atomic terms into complex terms), and finally formulas are defined in terms of operators which introduce statements (stand for logical connectives and predicates). An inductive syntax definition typically closes with a statement that no expressions other than the ones introduced are licensed in the given formal system. The language of first order predicate logic, the simplest language suitable for representing mathematics, may be formulated as follows:

The set of symbols consist of (countably infinite) sets of:

constants	$(7, \pi, \frac{13}{27}, \bot, \ldots)$
individual variables	$(x, y, z, x', x'', A, B, C, \ldots)$
n-ary functions	$(+, -, cos, \cup, \ldots)$
n-ary predicates	$(<, \subseteq, =, \ldots)$
logical connectives	$(\vee, \wedge, \Rightarrow, \ldots)$
quantifiers	$(\forall, \exists)$

The set of atomic terms consists of all constant and individual variable symbols.

If t_1,. . . ,t_n are terms and f is an n-ary function, then $ft_1. . . t_n$ is a term.

If t_1,. . . ,t_n are terms and P is an n-ary predicate, then $Pt_1. . . t_n$ is an atomic formula.

If $\mathbf{A}$ and $\mathbf{B}$ are formulas and x is a variable, then $\sim\mathbf{A}$, $\mathbf{A} \Rightarrow \mathbf{B}$, $\mathbf{A} \vee \mathbf{B}$, $\mathbf{A} \wedge \mathbf{B}$, $\mathbf{A} \Leftrightarrow \mathbf{B}$, $\exists x\mathbf{A}$, $\forall x\mathbf{A}$ are formulas.

[8] Boldface type preserved from the original.

```
V = { <IND_VAR>,<SET_VAR>, <SET_CONST>, <SET_FUNC>, <MEMB_PRED>, <SET_PRED>
      <OPEN_PAR>, <CLOSE_PAR>, <VRTCL_BAR>, <TERM>, <FORMULA> }
T = { x, y, z, x₁, . . . , A, B, C, . . . , ∅, ∩, ∪, \, . . . , ∈, ⊆, =, . . . , [, ], | }
S = set_expression

P :
<IND_VAR>         ::= x | y | z | x₁ | . . .
<SET_VAR>         ::= A | B | C | . . .
<SET_CONST>       ::= ∅
<SET_FUNC>        ::= ∩ | ∪ | \ | . . .
<MEMB_PRED>       ::= ∈
<SET_PRED>        ::= ⊆ | = | . . .
<OPEN_PAR>        ::= {
<CLOSE_PAR>       ::= }
<VRTCL_BAR>       ::= |

SET_EXPRESSION    ::= <TERM> | <FORMULA>
<TERM>            ::= <SET_VAR> | <SET_CONST> | <TERM> <SET_FUNC> <TERM> |
                      <OPEN_PAR> <IND_VAR> <VRTCL_BAR> <FORMULA> <CLOSE_PAR>
<FORMULA>         ::= <TERM> <SET_PRED> <TERM> | <TERM> <MEMB_PRED> <TERM>
```

Figure 3.1: Fragment of a context-free grammar for set expressions

The syntax of symbolic mathematical expressions, at least of their considerable subset, can be described in terms of context-free grammars (CFG).[9] A CFG for a subset of set theory expressions is shown in Figure 3.1.[10] The productions generate well-formed, however, structurally ambiguous expressions such as $A \cap B \in A \cap B \cup C$. $7 * 7 + 7$ is an analogous structure from arithmetics (neither set union and intersection nor addition and multiplication are associative). These kinds of structural ambiguities in mathematical expressions are common, however, they are immediately resolved based on the assumptions about *conventional* operator precedence (see Section 3.2.1.4). Grouping parentheses, which are part of the grammar, can be used to explicitly delimit ambiguous expressions, especially if non-default interpretation is intended.

[9] A context-free grammar, G, is a tuple (V, T, P, S), where V and T are finite sets of variables and terminal symbols, respectively, P is a finite set of productions of a form $A \rightarrow \alpha$ (with $A \in V$ and $\alpha \in (V \cup T)^*$), and S is the start symbol. Context free languages, generated by context-free grammars, were invented independently by Chomsky and Backus in the 1950s; the general idea dates back to Post's work on string rewriting production systems in the 1920s. Already Backus observed that algebra expressions can be analysed in terms of context-free grammars, while Wells (1961) and Anderson (1977) were among the first to apply the formalism in computational analysis of mathematical expressions. Fateman points at context-sensitive semantics of mathematical expressions and argues for the need of a more expressive formalism (Fateman, n.d.a); see also (Fateman, n.d.b).

[10] The grammar is presented in the Backus-Naur form. The abbreviated rule names for the terminal symbols stand for individual variables (IND_VAR), set variables (SET_VAR), set constants (SET_CONST), set functions (SET_FUNC), the membership predicate (MEMB_PRED), set predicates (SET_PRED), and opening/closing parentheses (OPEN_PAR/CLOSE_PAR). The

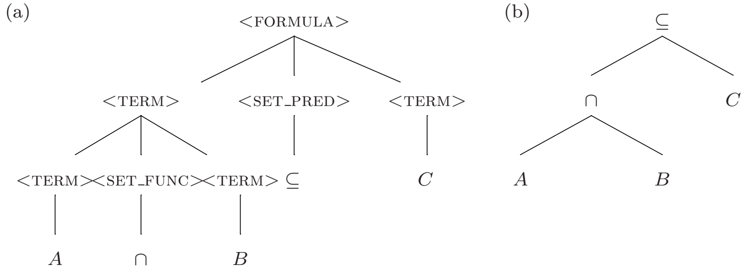


Figure 3.2: Tree representations of a mathematical expression; (a) Chomsky-style tree generated by the context free grammar in Figure 3.1, (b) head-daughter dependency-style

Internal structure Symbolic mathematical expressions can be represented as derivation trees of the CFG fragments that generate them. These trees correspond to phrase structure trees of natural language sentences and represent hierarchical constituency of the expressions' internal structure. For instance, based on the grammar in Figure 3.1, the set expression $A \cap B \subseteq C$ can be represented as shown in Figure 3.2 on the left. The three's nodes are labelled with the names of production rules and leaves are the terminal symbols (symbols from the vocabulary of the context-free language). The tree on the right represents the same expression in another diagrammatic presentation, with the operators at the tree-internal nodes and the operands at the leaves. This representation emphasises the relational nature of the operators and the recursive properties of the hierarchical structure of mathematical expressions: each complex expression has one main operator,[11] the root of the tree, and any number of atomic or complex subconstituents, subformulas, and subterms, which, in turn, can be identified by their main operator nodes and by tracing the subtrees headed by those nodes.[12] Note that some elements of the (sub-)structures may be omitted. We will return to this when we discuss *underspecification*.

vertical bar, |, denotes alternative productions. The grammar is obviously oversimplified (it does not, for instance, make a distinction between sets of different order: sets vs. sets of sets); it is meant only as an illustration.

[11] Chains of like terms, for instance, in iterated equations or in set expressions, such as $A \cup B \cup C \cup D$, can be thought of as right branching trees with the first operator in the chain as the root.

[12] There is empirical evidence that both experienced mathematicians as well as learners perceive mathematical expressions in terms of their syntactic structure, that is, our internal representation of mathematical expressions is based on the phrasal structure of the expressions' parse trees (Kirshner, 1987; Jansen et al., 1999, 2000, 2003).

Written notation Mathematical expressions written down on paper, a blackboard or rendered on a computer screen are of two-dimensional character. The vertical dimension is manifested, for instance, in the notation of fractions: the numerator is written above the denominator, the vertical structure emphasised by the fraction bar. Similarly in the notation for integration, limits, and iterated sum and product, the bounds are written above and below the operator symbol.

Along the horizontal dimension, symbolic expressions are linearised in a certain order. An interesting property of the internal tree structure of mathematical expressions is that they may be presented in different linearisation variants; much like word order in natural language. An expression can be written in the *infix notation* (operators linearised between operands they act upon), in the *Polish notation*, also known as prefix notation (operators precede the operands), or in the *inverse Polish notation* (operators follow the operands).[13] While there is a consistency in modern Western mathematics to linearise expressions with binary operators in the infix notation, there is little consistency in linearisation of different unary operators: the factorial symbol, !, is postposed with respect to its operand, the negation symbol, $\sim$ or $\neg$, preposed, the root symbol, $\sqrt{}$, preposed, in the notation for derivatives, the prime, $'$, is postposed, while d and ∂ preposed, powers of trigonometric functions may be either infixed (sin^2x) or postposed ($(sin\ x)^2$), etc. There is a special compact infix notation for writing down a series of formulas in a chain. If the relation between the objects is transitive, the terms can be iterated in a sequence: $\ldots = \ldots = \ldots$; similar notation is common for implication ($\Rightarrow$) and equivalence ($\Leftrightarrow$). A variant of the chain notation can occur with dual relations (for instance, $\ldots < \ldots > \ldots$ or $\ldots \subset \ldots \supset \ldots$).

The hierarchical internal structure, linearisation convention, and explicit delimitation of certain subexpressions give rise to a number of visually salient subparts of symbolic mathematical expressions which can be identified by their spatial location or marked delimitation. First, the horizontal dimension comes with the left- and rightwards orientation with respect to a certain point (or vertical line) of reference: the root of an expression's (sub-)tree (see Figure 3.2b). Second, the vertical dimension comes with the up- and down-ward orientation with respect to a certain horizontal line (or point) of reference: the topographic centre-line of a (sub-)expression in the linearised form (for instance, the fraction bar or a line running through the centre of an iterated summation symbol).[14] Third, due to marked delimitation, bracketed

[13] Paired symbols written on both sides of an expression (such as parentheses or absolute value vertical bars) are said to be in an *outfix/circumfix* or *mixfix/tranfix* notation.

[14] Mathematical expressions' topographic properties of this kind are exploited in mathematical OCR; see, for instance, (Fujimoto et al., 2003; Tapia and Rojas, 2004).

expressions also form distinguishable objects which, in turn, may embed other bracketed expressions.

Now, the purpose of this and the previous section, in which we illustrated the internal structure of mathematical expressions and their written form, is to lead up to a later discussion on referring in Section 3.2.2.5. Visually recognisable forms in mathematical notation give rise to a range of natural language spatial expressions which can be used to refer to the respective subparts of mathematical notation, exploiting its internal tree- and spatial structure and the relative location of its elements. We can, for instance, identify a term to the left of the main operator of an expression and refer to it as ''the left term'', ''the term on the left-hand side'', or ''the left side'' (keeping in mind the internal tree structure of the expression)[15] or identify a term enclosed in parentheses to the right of the main operator and refer to it as ''the term in brackets on the right'' or ''the right bracket''. In a language interpretation architecture, referents of these expressions need to modelled. We will return to this in Section 6.3.

Verbalisation Aside from referring to salient parts of notation, as exemplified above, we also read symbolic expressions out loud. Vocalisation routinely accompanies writing in a form of think-aloud (for instance, at the blackboard) or internal monologue. In mathematical textbooks, examples of natural language verbalisation may accompany introduction of new symbolism, as in the paragraph on set membership notation, cited earlier in this section, illustrates (see p. 96). In mathematical articles, a comment on wording may accompany introduction of new notion which the given article defines for the first time. Wording of ''known'' concepts is rarely explicitly stated in textbooks and certainly never in articles.[16] Learners must simply sooner (or later) ''pick it up'' in the classroom on their own. It is useful to realise that in the tutoring context this may result in misconceptions as to how symbolic expressions should be meaningfully read. Booker (2002) discusses difficulties that learners experience in understanding mathematics as a result of inconsistencies in the language used to talk about mathematics, especially its symbolism, and as a result of the fact that the verbal language bears no connection to the symbolic language used to record mathematical facts. Likewise, Thompson and Rubenstein (2000) stress the importance of teaching how to verbalise mathematics

[15] ''Left'' and ''right'' make sense with infix operators; the referring expression ''the left side'' fails in the context of $\sum n$, but succeeds in the context of $\sum n + m$. Referring expressions of this kind may also introduce ambiguities. Consider, for instance, ''the left side'' in the context of $\sum n + m = \sum m + n$.

[16] ''Known'' in inverted commas because what is assumed to be known is often left implicit...

and even suggest vocalisation of symbolic notation as one of oral strategies in teaching.[17]

While we are not aware of systematic studies addressing the linguistic structure of symbolic expressions spontaneously verbalised by expert mathematicians or learners,[18] it appears that in many cases, verbalisation of symbolic expressions follows the rules of syntax of the natural language in question, whereas the syntactic structures used in verbalisation reflect the object or proposition status of the entity which the expression denotes. Hence, terms (objects) are verbalised using noun phrase syntax, while formulas (propositions) using verb phrases.[19]

There is often more than one way of verbalising a given symbolic expression. For instance, the symbol for a function of one variable, x, written as $f(x)$ can be verbalised in English as a bare noun phrase ''f of x'' or simply ''f x'', a function of two variables, x and y, written as $f(x, y)$ can be verbalised as ''f of x and y'' or ''f of x y'', etc. Arithmetic expressions can be verbalised in different ways bringing out their process or concept nature. The term $2 + 2$, for instance, can be verbalised as a cardinal number, ''two plus two'' (with the word ''plus'' in the function of preposition, ''two, with two added'') or as co-ordinated cardinals, ''two and two'' (with the conjunction, ''and'', conveying

[17] Thompson and Rubenstein mention an example of a misconception about reading the logarithm notation which surfaced only by coincidence when a student in the class actually read an expression $log_2 8$ out loud as ''log of two to the eighth''. They refer to Usiskin who argued that ''[i]f a student does not know how to read mathematics out loud, it is difficult to register the mathematics'' (Usiskin, 1996, cited in (Thompson and Rubenstein, 2000)).

[18] But see (Karshmer and Gillan, 2003; Gillan et al., 2004) for a cognitive psychological study on understanding key issues in reading and understanding mathematical equations.

[19] There is a number of studies addressing speech interfaces for mathematical notation in the context of voice navigation in scientific documents and in the context of access to mathematics for the visually impaired. Since Raman's pioneering work on ASTER (Raman, 1994, 1997) there has been growing interest in various aspects of spoken interfaces for mathematics. (See, for instance, (Stevens et al., 1996; Fateman, n.d.a; Guy et al., 2004; Ferreira and Freitas, 2004; Fitzpatrick, 2002, 2006; Fateman, n.d.b) and references therein.) Pontelli et al. (2009) survey (multi-modal) accessible mathematics. Existing speech-enabled systems include MathTalk, MathSpeak, MathGenie MathPlayer, LAMBDA, AudioMath, TalkMaths. Fateman, among others, discusses a number of problems related to vocalisation of symbolic mathematical expressions, in general, however, studies aimed at accessibility necessarily tend to focus on wording which conveys the semantics unambiguously, independently of whether the proposed wording would be actually spontaneously produced by humans. Unique interpretation is ensured, among others, by special ''lexical indicators'', keywords which signal grouping. For instance, the expression $(a+b)/(c+d)$ might be verbalised as ''a plus b all over quantity c plus d'', where ''all'' signals the end of a term, ''over'' is short for ''divided by'' and ''quantity'' signals a start of a new grouping (Fateman, n.d.b). Fitzpatrick (2002; 2006) argues for effectiveness of speech prosody and standardised prosodic effects; see (O'Malley et al., 1973; Streeter, 1978; Stevens et al., 1996; Ferreira and Freitas, 2005) for investigation of prosodic correlates of mathematical expression structures.

aggregation). The equality symbol can be verbalised as the verb ''equal(s)'' or with a copula construction (''be'' in the sense of identity) or using action verbs, such as ''make'' or ''give'', which bring out the *process–concept* duality of the symbolic language (see (Sfard, 1991; Tall, 2004b)). The specific worded realisation depends on context (the term $2 + 2$ in isolation or within running text is not likely to be realised as ''two and two'', but rather as ''two plus two'', whereas in an equation both phrasings are possible, as in $2 + 2 = 4$.).

Aside from valid syntactic structures, symbolic expressions are sometimes verbalised using irregular syntax.[20] There is a range of symbolic forms which can be verbalised using idiosyncratic syntax which does not correspond to their internal structure. In English, arithmetic expressions can be worded as instructions (commands) in imperative mood. For instance, $2 + 2 - 1 = 3$ can be realised as ''two add two take away one leaves three'', which basically comprises four ellipted utterances (''(To/We have) two (objects/items), add two (objects/items), remove one (object/item),...'') Another class of irregularities comprises ungrammatical utterances. In English, this can be illustrated with the verbalisation of set expressions, for instance, ''A union B equals B union A'' for $A \cup B = B \cup A$. With ''A'' and ''B'' treated as proper noun categories, and ''union'' as a common noun, the structure ''A union B'' is ungrammatical, yet such constructions are routinely used to read expressions of this form. Examples of language artefacts related to irregular syntax in vocalisation which occurred in our corpora will be shown in Section 3.2.2.3.

3.2.1.3 Semantics

However formalised, mathematical expressions are often written in an underspecified way.[21] Omission of information may lead, in turn, to ambiguity. Classical lexical ambiguity is also found in mathematical language. In the following, these phenomena and the role of context and convention in disambiguation are briefly discussed.

Underspecification Frequently occurring forms of underspecification in the symbolic notation are *omission of notation elements* and *suppression of parameters* both of which can be explained in pragmatic terms as adherence to Maxims of Quantity and Manner in mathematics (see Section 3.3).

[20] Only two examples are shown here; data collection would be needed for a systematic analysis of the phenomenon.

[21] By *underspecification* we mean here omission of information, rather than underspecified semantic representation in a technical sense.

Table 3.1: Examples of ambiguous symbols, (a), and alternative notational conventions, (b)

(a)		(b)	
$\supset$	superset	‘‘A is a proper subset of B’’	$A \subset B$
	proper superset		$A \subsetneq B$
	implies		
$=$	number, set, function equality	‘‘A is a subset of B’’	$A \subset B$
	index assignment (as in $\sum_{n=0}^{\infty}$)		$A \subseteq B$
	name assignment ($f(x) = x^2 + 1$)		
(x, y)	open interval	‘‘p implies q’’	$p \Rightarrow q$
	ordered pair		$p \rightarrow q$
	inner product		$p \supset q$
	single-dimensional vector		Cpq

Delimitation symbols, in particular, brackets are one type of commonly omitted notation elements. From a formal point of view, unbracketed expressions may be considered syntactically ambiguous; the expression $2 + 2 * 2$ could be (in principle) interpreted as another name either for 8 or 6. This kind of underspecification is, however, typically immediately resolved based on assumptions on operator precedence. While operator precedence is rarely explicitly stated, in some domains (for instance, basic arithmetics) it is considered ‘‘common knowledge’’, an obvious part of general *conventions* in the given domain (discussed further in Section 3.2.1.4).

Wells (2003) points out another common type of underspecification in the symbolic expressions: suppression of arguments (parameters) of certain types of operations. An obvious example of suppression of parameters is the notation using primes for derivatives of functions of one variable. Indexed sums or products are often written with imprecisely specified summation bounds, however, in many cases, the omitted parameters are either explicitly stated in the natural language text surrounding the symbolic expression or can be inferred from it. For instance, if in a given paragraph or section n is declared to be a natural number, an underspecified expression $\sum_n$ can be interpreted as $\sum_{n=0}^{n=\infty}$ or $\sum_{n=1}^{n=\infty}$, depending on whether the adopted convention is for the set of natural numbers to include 0 or not.

Ambiguity Ambiguities in the symbolic language result from the fact that mathematical symbols are often *polysemous*. One symbol may denote different objects depending on the context in which it is used, in particular, on the subarea of mathematics in question; this can be considered a special case of *lexical ambiguity* in mathematical language.

The omnipresent equality sign, $=$, is a notorious example of a polysemous symbol. Depending on context, the equality sign takes different types of operands as arguments and is interpreted accordingly.[22] Object naming symbols, certain punctuation, and typographical layout have the same property; for instance, the dot may occur as the multiplication symbol, the decimal separator, or as punctuation separating the bound variable(s) and the body in a quantified formula, a superscripted number may be interpreted as a power operator (2^2, x^2), except in the context of functions, where it may denote the n-th derivative ($\frac{d^2F(x)}{dx^2}$), unless it is a -1, in which case it is an inverse function (f^{-1}), unless, of course, it is indeed an exponent ($(sin\ x)^{-1}$). Even special layout elements can be polysemous; consider the horizontal bar in $\frac{7}{13}$ vs. $\frac{dy}{dx}$. Table 3.1a shows other examples of polysemous notation and their interpretations.

Given the abundance of polysemy, it is no wonder that learners struggle with notation (Moore, 1994; Dorier et al., 2000; Downs and Mamona-Downs, 2005). However, an experienced reader can in most cases disambiguate the symbolic notation instantaneously using context and his knowledge of mathematical conventions.

3.2.1.4 Conventions and context

The use and the interpretation of the so-called ‘‘formal’’ mathematical language is to a large extent governed by convention and the mathematical context. Although in principle any symbol can be defined to denote any object (for instance, the symbol A could be declared to stand for the subset relation) certain traditional conventions are generally followed and the knowledge of these conventions is assumed of the recipient of a mathematical text.

By convention, certain symbols have fixed interpretations ($\aleph_0$, ∞, $\emptyset$, or the Arabic and Roman number symbols), while others systematically evoke preferred readings in specific contexts (π, e, $\Re$, $\sum$, $\prod$, ϵ, δ, i, etc.). Objects of certain types are typically denoted by specific symbols. For instance, functions are typically denoted by the primed, sub- or super-scripted letter f, groups by uppercase G, relations by uppercase R (following the mnemonic convention), summation index variables by n or i, and sets by uppercase letters from the beginning of the alphabet. Also by convention, functionally related objects tend to be denoted by the same letter names distinguished by accents (circumflex, check, tilde, bar) or primes ($\hat{X}$ might be chosen to name the

[22] Multi-purpose use of operators corresponds to function or method *overloading* in programming.

closure of X), upper-case letters tend to be used for structures (structured mathematical objects) and lower-case letters for the elements of structures, primes are used to mark collections of objects of the same type (x', x'',. . . for the elements of a set X), and stylised letter shapes and typefaces for specific distinguished objects (blackboard bold style or German Altschrift, fraktur, for specific number sets: reals, integers, complex).

The choice of symbols itself is also a matter of convention. For instance, the subset relation is denoted as $\subset$ by some authors and as $\subseteq$ by others, open/closed intervals may be denoted as $(.,.)/[.,.]$ or as $(.,.)/< .,. >$, the cardinality of a set S as $K(S)$, $K(K(S))$, $\|S\|$, etc. National and cultural conventions may differ; for instance, in Western Europe and North America, the symbols $\exists$ and $\forall$ are used for the existential and the universal quantifier respectively, while in Eastern Europe $\bigvee$ and $\bigwedge$ are still used, although the Western convention tends to take over. Also, different conventions are applied in mathematics and in natural sciences or engineering; for instance, in algebra vectors are denoted by boldface letters from the end of the alphabet ($\boldsymbol{x}$) while in physics the arrow notation is common ($\vec{V}_x$ for the x-component of a velocity vector), the imaginary part of a complex number is denoted with i in maths and typically with j in engineering.[23] Table 3.1b shows other examples of notational alternatives.

Knowledge of mathematical conventions plays a role in interpreting symbolic notation, in particular, in interpreting expressions which appear ambiguous. Already in elementary arithmetics we are taught that multiplication should be performed before addition, hence, the expression $2 + 2 * 2$ can be unambiguously interpreted without parentheses. This interpretation exploits the notion of *precedence* among operators, that is, rules that state which operators must be applied first or which operators have ‘‘higher’’ and which ‘‘lower’’ precedence.[24]

Finally, interpretation of the symbolic notation depends on context, both the *textual context* as well as the *mathematical domain context* in which the given notation is used. For instance, in the context of binary relations, (x, y) is not likely to denote an interval and in the context of complex numbers, the lowercase i is reserved for the imaginary part of a complex number and when a summation index over complex numbers is used, it should be different from i. Similarly, concatenation is interpreted with respect to context;

[23] See (Libbrecht, 2010) for further examples and (Kohlhase and Kohlhase, 2006) for a discussion on communities of practice in mathematics and implications for representing notation.

[24] A thorough precedence table can be found on the Mathematica website: `http://reference.wolfram.com/language/tutorial/OperatorInputForms` [Accessed: 2007].

while 77 denotes a natural number, $7x$ typically denotes multiplication, $3\frac{1}{2}$ addition, whereas $\sin x$ functional application.

3.2.1.5 Errors in the symbolic language

Learning the language of mathematics, much like learning a foreign language, involves making mistakes. Therefore, it is not surprising that symbolic expressions produced by students are prone to errors, both of form and substance. While texts written by mathematicians contain only valid and pertinent statements (or at least, published mathematics, should contain only those), learners' discourse may contain statements that are false or irrelevant in the given context. These are errors of substance, of *pragmatic* nature. Diagnosing and addressing these types of errors requires knowledge beyond the mere knowledge of the symbolic language, namely, the knowledge of the given domain, the ability to reason within this domain and, in the case of tutorial dialogue, the knowledge of pedagogical criteria (for instance, what is an appropriate size of a proof step from a pedagogical point of view).

In general, before a semantic and pragmatic evaluation of a symbolic expression can take place, the expression must be ascertained to be meaningful in the given symbolic language. An expression is *well-formed* when it conforms to the rules of syntax for expressions from the given mathematics subarea or to the rules of admissible simplified presentations (for instance, rules which permit to reduce the number of parenthesis without introducing ambiguity; we mentioned this already in Section 3.2.1.1.) Well-formedness concerns an expression's structure, its syntax and the properties of the lexical identifiers of which it is composed.

There is a range of errors affecting the form of mathematical expressions which render them *ill-formed* and thereby meaningless. Unlike mathematical textbooks and research publications, in which most errors of form can be most likely attributed to unfortunate typographical oversight and only rarely to misconstrued reasoning or lack of knowledge, students' writing may contain errors which are due to genuine misconceptions. Moreover, computer-based mathematics can be additionally error-prone due to keyboarding or interface problems. Students' input may be especially affected in this respect because the blackboard and paper still remain the primary media for written mathematics up to the level of university education.

Generally speaking, errors of form in the symbolic language can be categorised into two broad classes of *structural* and *semantic* errors. Structural errors affect the syntactic structure of mathematical expressions, while semantic errors affect their semantic interpretation.[25] Expressions with structural errors cannot be parsed by a normative grammar for terms and formulas in the given domain. Expressions with semantic errors, while structurally valid, cannot be assigned a meaningful interpretation or, in case of truth-valued expressions, are simply false. A well-formed and semantically meaningful proof step may be still inappropriate for *pragmatic* reasons: it may be irrelevant for the given task or, even if relevant, it may be too much of an ‘‘argumentative shortcut’’, too large a step. Pragmatic errors arise at the level of proof steps (rather than individual symbolic expressions) and in the given proof discourse context.

An analysis of the two corpora revealed a number of further subcategories of form errors produced by learners. Among structural errors there are two subcategories: *Segmentation* errors are possibly an artefact of keyboard input and are due to omitting white-space or punctuation (in the notation for pairs, (sr) in place of (s, r), for instance) resulting in fused identifiers. *Delimitation* errors arise from inappropriate use of parentheses: opening or closing parenthesis are omitted (*Parenthesis mismatch*), both parentheses are omitted in a term which requires bracketing (*Missing parentheses*), or double (or more) unnecessary parentheses are used (*Superfluous parentheses*). Finally, a constituent, atomic or complex, may be omitted resulting in a *Constituent structure* error corresponding to invalid predicate–argument structure in natural language. Among semantic errors, a distinction can be made between lexical errors and correctness errors. Lexical errors arise from inappropriate use of identifiers: an expression may contain an identifier which has not been defined in the given context (*Unknown identifier*) or a known identifier is used inappropriately (*Inappropriate identifier*). As a result of the latter an expression becomes *ill-typed*: some of the expression's operators are applied to incompatible operands; this corresponds to a violation of sortal restrictions in natural language. Correctness errors have to do with validity of truth-valued expressions. The two subclasses of pragmatic errors have to do with relevance and granularity of proof steps. An overview of the error categories is shown in Table 3.2.[26]

[25] While we are not aware of systematic studies dedicated solely to form errors in the symbolic language, there is a number of related studies in the larger context of mathematics learning disabilities; see (Magne, 2001) for an extensive bibliography on special educational needs in mathematics and also, for instance, (Kennedy et al., 1970; Hall, 2002; Melis, 2004) for error patterns in problem solving in general.

[26] The classification summarises only observations based on the two collected corpora. Thus, it is not meant as exhaustive. Our preliminary error categorisations were presented in (Horacek and Wolska, 2005b, 2006b) and issues related to generating responses to erroneous statements in (Horacek and Wolska, 2007, 2008).

Table 3.2: Categories of errors in students' mathematical expressions

Error category	Description	Code
Structural errors	Expression ill-formed	I
Segmentation	Omission of white-space or punctuation	I-1
Delimitation	Inappropriate use of grouping symbols	I-2
Parentheses mismatch	Opening or closing parenthesis missing	I-2-a
Missing parentheses	Required parentheses omitted	I-2-b
Spurious parentheses	Extra parentheses	I-2-c
Constituent structure	Constituent missing	I-3
Semantic errors	Incorrect or unknown identifiers or invalid statement	II
Unknown identifier	Identifier not defined in context	II-1
Wrong identifier	Known identifier used incorrectly	II-2
Correctness error	False statement	II-3
Pragmatic errors	Logical argument invalid or inappropriate	III
Relevance error	True expression unrelated to solution	III-2
Granularity error	Inappropriate proof step size	III-3

Table 3.3 shows examples of flawed expressions from C-I and C-II and their corresponding error categories given the identifiers defined for the proof exercises in the experiments.[27] Examples (e1)–(e5) illustrate structural errors. In (e1) not only a space between the operator symbol P and the identifier C, but also the parentheses required for the powerset operator are missing; as a result, the token PC is an unknown identifier (lexical error). The expression (e2) is incomplete (closing bracket missing), (e3) is structurally ambiguous because the required brackets have been omitted, whereas in (e4) duplicate brackets are unnecessary. In (e5) the second constituent in the pair object is missing. Examples (e6)–(e17) illustrate semantic errors. The lexical errors in (e6) are most likely due to sloppy keyboarding: not only are the set identifiers a and b in the wrong case, but also the symbol p is used in place of the set identifier B; even if we accepted the lower-case symbols as a typos, p would still be an example of inappropriate identifier use (operator in place of a variable). In (e7) undeclared variables, x and y, are used even though a previous declaration was made for the given context, b and a. Examples (e8)–(e11) illustrate the

[27] Defined symbols were: A, B, C, M for first order sets, R, T, S for relations, x, y, z for individual variables, P for the powerset of a set, K for set complement, and $^{-1}$ for the inverse relation, as well as basic naïve set theory and predicate logic symbols. Erroneous symbols are boxed; empty boxes denote omitted symbols. Previous context, where relevant, is shown in square brackets. Error codes refer to Table 3.2.

Erroneous expression	Error code
(e1) $P((A \cup C) \cap (B \cup C)) = \boxed{PC} \cup (A \cap B)$	I-1, I-2-b, II-1
(e2) $\exists z \in M : ((x, z) \in R \wedge (z, y) \in T) \vee ((x, z) \in S \wedge (z, y) \in T\boxed{}$	I-2-a
(e3) $(a, b) \in \boxed{}R \circ T\boxed{} \cap \boxed{}S \circ T\boxed{}$	I-2-b
(e4) $(R \cup S) \circ T = \boxed{(}((R \circ T) \cup (S \circ T))\boxed{)}$	I-2-c
(e5) $S^{-1} \circ R^{-1} = \{(x, y) \mid \exists z(z \in M \wedge (x, \boxed{}) \in S^{-1} \wedge (z, y) \in R^{-1})\}$	I-3
(e6) $(\boxed{p} \cap \boxed{a}) \in P(\boxed{a} \cap \boxed{b})$	II-1
(e7) $[(b, a) \in (R \circ S), z \in M] \ldots (\boxed{x}, \boxed{z}) \in R$ und $(\boxed{z}, \boxed{y}) \in S$	II-1
(e8) $(x \in \boxed{b}) \boxed{\notin} A$	II-1, II-2
(e9) $A \subseteq K(B)$ then $A \boxed{\notin} B$	II-2
(e10) $[M : \text{set}] \ldots (x, y) \boxed{\in} M$	II-2
(e11) $x \boxed{\subseteq} K(A)$	II-2
(e12) $(T^{-1} \circ S^{-1})^{-1} \cup (T^{-1} \circ R^{-1})^{-1} \boxed{\Leftrightarrow} (y, x) \in (T^{-1} \circ S^{-1}) \vee (y, x) \in (T^{-1} \circ R^{-1})$	II-2
(e13) $(R \cup S) \circ T = \{(x, y) \mid \exists z(z \in M \wedge (x, z) \in \{\boxed{x} \mid \boxed{x} \in R \vee \boxed{x} \in S\} \wedge (z, y) \in T)\}$	II-2
(e14) $\exists z \in M : (x, y) \in R \circ T \vee (x, y) \boxed{\vee} S \circ T$	II-2
(e15) $(R \circ S)^{-1} = \{(x, y) \mid \exists z(z \in M \wedge (y, z) \in R^{-1} \wedge (z, x) \in S^{-1})\} \boxed{\subseteq} S^{-1} \circ R^{-1}$	II-3
(e16) $P((A \cap B) \cup C) \boxed{=} P(A \cap B) \cup P(C)$	II-3
(e17) $[(s, r) \in (R \circ S)^{-1}] \ldots \boxed{(s, r)} \in R \circ S$	II-3

Figure 3.3: Examples of invalid symbolic expressions from students' proofs

common confusion of the subset and membership relations on sets. In (e8) there is additionally an unknown symbol b. In (e10) the student appears to think that M contains pairs (is a relation) whereas M was declared as a set in the task definition. A type mismatch arises due to a wrong operator in (e11) and (e12). In (e13), the same variable, x, is used in two contexts in which it would have to be of different types: first as an element of a pair and then as an element of a set. In (e14) unrelated operators have been confused: $\vee$ in place of $\in$. (e15)–(e17) are examples of logically incorrect statements: in (e15) and (e16) a stronger and weaker assertion, respectively, is expected (about equality of sets rather

Table 3.3: Possible sources of symbol confusion and the resulting errors

Possible error source	Examples of confused symbols	Error category
Dual operator	$\{\subseteq, \supseteq\}$, $\{\subset, \supset\}$, $\{\cap, \cup\}$, $\{\wedge, \vee\}$	II-3, III-2
Stronger/weaker relation	$\{\subset, \subseteq\}$, $\{\subseteq, =\}$, $\{\supset, \supseteq\}$, $\{\supseteq, =\}$	II-3, III-2
Conceptually related	$\{\subseteq, \in, \subset\}$, $\{\supseteq, \ni, \supset\}$, $\{\Leftrightarrow, =\}$	II-2, II-3, III
Typographical artefact	$\{\cup, \vee\}$, $\{\cap, \wedge\}$, $\{K, P\}$, $\{a, b\}$, $\{P, B\}$	II, III

than inclusion, or vice versa). A logical error in (e17) is caused by swapped variables. Among pragmatic errors, shown in Figure 3.4,[28] (1) illustrates a step which the tutor considered irrelevant (definition instantiation in S20) and (2) and (3) are step size errors: in (2) the student restates the proposition to be proven, an open goal, in his second step (too coarse-grained) and in (3) the tutor considered spelling out the definition unnecessary (too detailed). As mentioned previously, pragmatic errors are of different nature than structural and semantic errors; recognition of these errors involves not only reasoning but also pragmatic criteria, for instance, pedagogical criteria stemming from the adopted pedagogical strategy and the student model.

A closer look at the most common erroneous expressions reveals a pattern within the class of semantic errors which may be due to systematic misconceptions that students have about pairs of set theoretic and logical operations. A subclassification of semantic and pragmatic errors with respect to their possible source is shown in Table 3.3. Often recurring errors result from students confusing operators which are ''dual'', in a broad sense of the word, with respect to each other. Examples of these include the logical conjunction and disjunction (dual with respect to negation), the set union and set intersection (dual with respect to set complement; analogous to the former), and (partial) order relations on sets (subset vs. superset); example (e14) in Table 3.3 illustrated erroneous conjunction in place of disjunction. Confusion about ordering relations results, moreover, in statements which are weaker or stronger than the expected statements, as in (e15) and (e16). A large number of errors have to do with confusion about the set hierarchy (sets vs. sets of sets) and the set membership and set inclusion relations which are conceptually related, as in (e8)–(e11). Misconceptions related to these concepts have been previously discussed by Zazkis and Gunn (1997) and Bagni (2006). Set equality and logical equivalence, as in (e12), are another pair of confusable relations; see, for instance, (Kieran, 1981; Sáenz-Ludlow and Walgamuth, 1998; Knuth et al.,

[28] Tutors' evaluations of correctness, granularity, and relevance of the steps are shown in angular brackets along with the corresponding categories from Table 3.2.

(1) S: Es bleibt zu zeigen: $(R \cup S) \circ S = (R \cup S) \circ S^{-1}$

T: ⟨correct, appropriate, relevant⟩
Ist diese Aussage allgemein gueltig?
Is this statement valid for the general case?

S: Ja.
Yes.

T: ⟨incorrect / II-3, appropriate, relevant⟩
Koennen Sie das beweisen?
Can you prove it?

S: Es sei $(x, y) \in (R \cup S) \circ S$, also gibt es ein $z \in M$ mit $(x, z) \in R \cup S$ und $(z, y) \in S$
Let $(x, y) \in (R \cup S) \circ S$, then there is a $z \in M$ such that $(x, z) \in R \cup S$

T: ⟨correct, appropriate, irrelevant / III-2⟩
Ihre Folgerung stimmt.
Your conclusion is correct.

(2) S: $(R \cup S) \circ T = (R \circ T) \cup (S \circ T)$

T: ⟨correct, appropriate, relevant⟩
Das ist richtig!
That's correct!

S: $(R \circ T) \cup (S \circ T) = (T^{-1} \circ R^{-1})^{-1} \cup (T^{-1} \circ S^{-1})^{-1}$

T: ⟨correct, too coarse-grained / III-3, relevant⟩
Wie kommen Sie darauf? Gehen Sie in kleineren Schritten vor!
How did you come arrive at this? Please use smaller steps!

(3) S: Wenn $(x, z) \in S^{-1}$ und $(z, y) \in R^{-1}$, dann gilt $S^{-1} \circ R^{-1}$
If $(x, z) \in S^{-1}$ and $(z, y) \in R^{-1}$, then $S^{-1} \circ R^{-1}$ holds

T: ⟨partially correct, too detailed / III-3, relevant⟩
Meinen Sie vielleicht $(x, y) \in S^{-1} \circ R^{-1}$?
Do you mean $(x, y) \in S^{-1} \circ R^{-1}$, perhaps?

Figure 3.4: Examples of proof steps inappropriate in terms of relevance and granularity.

2005) for a discussion on students' problems with equality and equivalence. The last group of errors, involving unrelated symbols, may be simply artefacts of typographic or shape similarity, or genuine typo or oversight errors.

What is interesting and relevant from the point of view of computational processing is that the tutors rarely rejected utterances with *Delimitation* errors, even if more than one was present:

(4) S: $\exists z(z \in M \wedge (((x,z) \in R \wedge (z,y) \in T)\boxed{)} \vee$
$\boxed{}(x,z) \in S\boxed{)} \wedge (z,y) \in T))) =$
$\exists z(z \in M \wedge (x,z) \in R \wedge (z,y) \in T)\boxed{)} \vee$
$\exists z(z \in M \wedge (x,z) \in S \wedge (z,y) \in T)$
T: ⟨correct, appropriate, relevant⟩
Bis auf Klammerung korrekt. Fahren Sie fort!
Correct up to bracketing. Go on!

Tutors accepted ill-formed steps of this type in 53 cases. Only in 7 cases did they explicitly request a correction. This means that tutors tended to focus on the proving task, rather than low-level syntactic details. Ideally, a cooperative system should behave analogously, which, in turn, means that it needs a robust parser for mathematical expressions. In Section 6.4 we present a preliminary study aimed at automated correction of errors of some categories.

3.2.2 The informal language

While the formal language of mathematics consists of symbolic expressions, the most prominent characteristics of the *informal* language is the familiar combination of natural language phrases and symbolic expressions, with symbolic expressions smoothly embedded into the natural language text. In this section we turn to this informal language.

3.2.2.1 Multi-modality

A typical sentence from a mathematical proof, be it in a textbook or in tutorial dialogue, may look, for instance, as follows:

(5) Wenn $x \in B$ dann $x \notin A$
If $x \in B$ then $x \notin A$

(6) $K(A \cup B)$ ist laut DeMorgan-1 $K(A) \cap K(B)$
$K(A \cup B)$ is by DeMorgan-1 $K(A) \cap K(B)$

(5) is a prototypical conditional statement. (6) states an equality between two sets and provides a justification. The equality is expressed with a predicate worded in natural language, ‘‘ist’’ (*is*), and two symbolic expressions, $K(A \cup B)$ and $K(A) \cap K(B)$, denoting sets. The justification is expressed in words using an adverbial construction, ‘‘nach + Dative’’ (*by*). While the equality could be stated with the equality sign, there is no standard symbolic notation for

justifications of proof steps in *narrative* mathematical text; justifications are thus signalled in natural language.[29]

In the tutorial dialogues in our corpora, this kind of embedding of symbols within natural language occurs also in variants which are rather not frequently found in textbooks or publications:

(7) $A \cap B$ ist $\in$ von $C \cup (A \cap B)$
$A \cap B$ is $\in$ of $C \cup (A \cap B)$

(8) Nach der Definition von $\circ$ folgt dann (a, b) ist in $S^{-1} \circ R^{-1}$
By definition of $\circ$ it follows that (a, b) is in $S^{-1} \circ R^{-1}$

(9) A auch $\subseteq B$
A also $\subseteq B$

In (7) and (8) the set membership symbol, $\in$, and relation composition symbol, $\circ$, have been used as a kind of shorthand for a part of the object of the main predicate, ''to be an element of'' or prepositional phrase ''(definition) of composition relation''. These examples illustrate two tendencies in informal mathematical discourse: one towards *natural language verbalisation* and the other towards a *telegraphic style*. The same sentence could be expressed more economically using a symbolic expression alone, yet wording is perhaps more natural. In (9) an additive adverb is verbalised within the formula. There is no symbolic notation corresponding to the intended meaning of ''auch'' (*also*), however, from the mathematical point of view, the adverb does not add any mathematical content, so it could be omitted altogether.[30]

The most interesting characteristic of the two language modes which form the informal mathematical language is that they are *complementary* and *interchangeable* with respect to each other: they can be flexibly interleaved, either one, the other, or both can be used to express the same mathematical content, and different parts of mathematical content can be expressed using one mode or the other. Examples (10) through (14) illustrate these properties:

(10) $x \in B \implies x \notin A$

(11) Wenn $x \in B$ dann $x \notin A$
If $x \in B$ then $x \notin A$

(12) B enthaelt kein $x \in A$
B contains no $x \in A$

(13) A hat keine Elemente mit B gemeinsam.
A has no elements in common with B.

[29] (Unlike in tabular presentations, such as Fitch-style natural deduction, in which rule names, typically abbreviated, are placed in a dedicated layout area, along with references to line labels.)

[30] We will return to the discussion of pragmatic aspects in mathematical discourse in Section 3.3.

(14) A enthaelt keinesfalls Elemente, die auch in B sind.
A contains no elements that are also in B

All the above utterances express the same content: the claim that the sets A and B are disjoint. They do this, however, using different language modes: (10) using symbols alone, (11) and (12) using mixed language, and (13) and (14) using natural language with only the set names expressed as symbols. The difference between (11) and (12) is in what is verbalised: the implication in (11) and the relation between the set elements in (12).[31] While in (11) the symbolic and natural language parts form independent constituents, there is a constituent overlap of a kind between the symbolic and natural language parts in (12): the scope of the worded negation ''kein'' (*no*) is only over x, which is a part of the symbolic expression following it. Similar interaction and textual context dependence can occur with other scope-bearing natural language word classes, such as (generalised) quantifiers (*all*, *every*, *any*, *only*, etc.) The scope of the overlap (that is, of the quantifier) depends on the semantic context. If B is a set whose elements are mathematical formulas, the expression $x \in A$ could be considered a mention of a particular element of this set. In this case the scope of negation would be over the entire expression. Constructions of this type can be found in textbooks and publications. (13) and (14) show that the same content can be naturally expressed using words alone with only atomic terms, set variables, expressed as symbols, and that various syntactic constructions can be employed. In (13), a complex predicate ''gemeinsam haben'' (*have in common*) is used; ''haben'' (*have*) is a kind of support verb here; the actual lexical meaning is expressed by the adverb ''gemeinsam''. In (14) a complex noun phrase with a relative clause post-modification is used.

Much like symbolic language can be fluently embedded within natural language, the opposite is also possible: natural language can be incorporated into symbolic expressions. This occurs when there would be no benefit of the symbolic presentation because the focus is not on the formalisation of the worded concept; that is, if the symbolic representation is not relevant and would only cause unnecessary additional cognitive load on the part of the reader. Consider for example the following expressions which introduce a certain number set:

$$A = \{\ p \mid p \in \mathbb{Z} \wedge \exists x \in \mathbb{Z},\ p = 2x + 1\ \}$$
$$A = \{\ p \mid p \in \mathbb{N} \wedge (\forall x \in \mathbb{N},\ \forall y \in \mathbb{N},\ p|xy \Rightarrow p|x \vee p|y)\ \}$$
$$A = \{\ p \mid p \in \mathbb{N} \wedge \neg\exists x \in \mathbb{N},\ \exists y \in \mathbb{N}\,(x < p \wedge y < p \wedge xy = p)\ \}$$

[31] A classification of proof contributions with respect to the type of content worded in natural language will be presented in Chapter 4 (Section 4.3.4).

$$\begin{aligned} A = \{\ p \mid p \in \mathbb{N} \wedge\ & \exists x \in \mathbb{N},\ p = x + 2 \\ & \wedge \neg\exists x \in \mathbb{N}, \exists y \in \mathbb{N},\ ((x+2) * (y+2) = p)\} \end{aligned}$$

and their counterparts in informal language with natural language wording:

$A = \{\ p \mid p$ is odd $\}$

$A = \{\ p \mid p$ is prime $\}$

Unless the purpose of these examples were to symbolically formalise the notions of an odd or a prime number, the natural language presentation of a familiar concept is preferred. These examples show that the symbolic notation, merited for its brevity and succinctness, is not always that brief. Hence, natural language wording is also preferred for concepts whose formalisation is difficult or complex. We will return to this and related issues when we discuss Gricean Maxims in mathematical discourse in Section 3.3. What all the examples in this section illustrate is that parsing symbolic expressions in the context of natural language surrounding them is a basic requirement that a computational interpretation module for mathematical language must fulfil.

3.2.2.2 Lexicon

The vocabulary of the mixed language of mathematics consists of the vocabulary of the symbolic notation and the vocabulary of natural language. The latter follows its own morphology and orthography rules. As illustrated above the two language modes can be tightly interleaved. The vocabulary of symbols may be used to substitute entire natural language phrases (π for ‘‘the ratio of the circumference of a circle to its diameter’’ or $\in$ for ‘‘is an element of’’/‘‘belongs to’’) which often do not even form linguistic constituents ($\forall$ for ‘‘for all’’, $\Leftrightarrow$ for ‘‘if and only if’’, or $\notin$ for ‘‘is not an element of’’). Mathematical symbols typically do not undergo linguistic inflectional processes in writing[32] other than acquiring genitive forms, as in ‘‘x’s value’’ or ‘‘A’s elements’’.

The lexicon of mathematical language consists of a subset of the lexicon of ordinary language, the *general lexicon*, and a terminological part specific to the mathematical domain, the *terminological lexicon*. In this respect, mathematical language is lexically more complex than everyday language.

Technical vocabulary Many mathematical words have Greek or Latin origin: ‘‘isosceles’’, ‘‘asymptotic’’, ‘‘idempotent’’, etc. There is also a set of lexemes coined as neologisms, for instance, ‘‘pathocircle’’, ‘‘polygenic’’,

[32] In verbalisation they do of course.

‘‘ultraradicals’’.[33] Some lexemes from the general lexicon acquire special technical meaning in the context of mathematics (meaning restriction or specialisation) and in most cases the new meanings are impossible to guess: the terms ‘‘group’’ or ‘‘field’’ are such examples. In the process of meaning specialisation, a common word may also obtain a new grammatical category, for instance, ‘‘integral’’: an adjective in the general lexicon, a noun in the mathematical terminology.[34] Thompson and Rubenstein (2000) discuss lexical phenomena in mathematical language from the point of view of potential problems which may arise during learning. Table 3.4 summarises a fragment of their classification.[35]

Multi-word lexical units A multi-word expression is a general term for different kinds of linguistic units consisting of two or more words, be it phrasal lexemes, phraseological units or multi-word lexical items. These include: named entities (names of places, persons, organisations, etc.),

[33] Examples from *Mathematics and the imagination* by Kasner and Newman.

[34] An interesting resource on the earliest uses of mathematical terminology is maintained at `http://jeff560.tripod.com/mathword.html` [Accessed: 2007]. Becker's work retraces the evolution of mathematical concepts in the 19th century and the changes in the terminology and the semantics of the language used (Becker, 2006).

A digression: A lot of mathematical terminology (technical terminology in general) in Western languages – English, German, and French – have the same etymological roots: Latin, Greek, or Arabic. (See (Schwartzman, 1994) for the origins of English mathematical terms.) By contrast, Polish terminology bears no resemblance to the Western counterparts: compare, for instance, ‘‘integral’’/‘‘Integral’’/‘‘intégrale’’ vs. ‘‘całka’’, ‘‘differential’’/ ‘‘différentielle’’/‘‘Differential’’ vs. ‘‘różniczka’’, or ‘‘derivative’’/‘‘dérivée’’ vs. ‘‘pochodna’’. A lot of the Polish terminology is due to Józef Jakubowski's translations of French works and Jan Śniadecki's contributions to popularising mathematics. Śniadecki believed that in order for mathematics to be accessible, it *should* use national terminology and the *vocabulary should be derived from common words by analogy with their use in known contexts* (Śniadecki, 1813).

[35] Only one example from each mathematical area is given. For further examples, see the original source. The category descriptions are reproduced as in the original text, except we do not refer to English since the phenomena are cross-linguistic. A simpler classification was previously proposed by Shuard and Rothery: Mathematical words are classified into three types: (i) technical words (those which have meaning only in mathematics; for instance, ‘‘square centimeters’’), (ii) lexical words (those which have a similar meaning in mathematics and in everyday language, for instance, ‘‘reminder’’, ‘‘origin’’), (iii) everyday words (those which occur both in everyday language, but can have both similar *and different* meanings in mathematics and everyday language, for instance, ‘‘points’’, ‘‘change’’); (Shuard and Rothery, 1984), as reported in (Raiker, 2002).

The importance of understanding the differences in word usage between everyday language and mathematical language in the process of learning mathematics has been also discussed in (Kane et al., 1974; Usiskin, 1996; Raiker, 2002), to mention just a few. Booker (2002) attributes the difficulties that children experience in mathematics to the inconsistencies in the language and a lack of connections between the way ideas are represented, the language to talk about them, and the symbols used to record them.

Table 3.4: Excerpt of Thompson and Rubenstein's (2000, p. 569) classification of lexical phenomena in mathematical language

Lexical phenomenon	Examples
Words shared by mathematics and everyday language, but with distinct meanings	prime, imaginary, right (angle), combination, tree
Words shared with natural language, with comparable meanings, the mathematical meaning being more precise	equivalent, limit, similar, average, and
Terms found only in mathematical context	quotient, asymptote, quadrilateral, outlier, contrapositive
Words with more than one mathematical meaning	inverse, base, round, range, dimension
Modifiers that change mathematical meaning in important ways	value vs. absolute value, root vs. square root, bisector vs. perpendicular bisector, number vs. random number, reasoning vs. circular reasoning
Idiomatic mathematical phrases	at most, one-to-one, if-and-only-if, without loss of generality

idioms (‘‘get off scot-free’’ and ‘‘Bob’s your uncle’’), phrasal collocations (‘‘make a claim’’, ‘‘take a stand’’), conventional metaphors (argument is journey: ‘‘follow an argument’’, argument is balance: ‘‘shaky argument’’, argument is war: ‘‘defend an argument’’), proverbs and sayings (‘‘As you saw, so shall you reap’’, ‘‘The truth will out’’, ‘‘Unless a miracle happens’’), similes (‘‘lie like a pro’’, ‘‘cunning as a fox’’), and routine formulae (‘‘you know what I mean’’, ‘‘beyond any doubt’’). We used the more general term ‘‘multi-word units’’ here, rather than ‘‘multi-word expressions’’, because the latter, under current interpretations, are typically associated with non-compositionality of meaning. Mathematical discourse is abound in multi-word units; some of which are non-compositional.

The obvious multi-word named entities, aside from numeric expressions, include names of theorems, lemmata, conjectures, hypotheses, and axioms, which are often named after the researcher who introduced them, for instance, ‘‘Peano’s Axioms’’. Named entities of this type often appear in different syntactic, lexical, and spelling variants, for instance, Peano’s Axioms are also known as ‘‘Dedekind-Peano axioms’’ or ‘‘Peano postulates’’, the name of De Morgan’s laws can also be referred to as ‘‘De Morgan laws’’ or ‘‘the laws of De Morgan’’.

The tutorial dialogue corpora contain numerous occurrences of multi-word names of set theory and binary relation theorems and lemmata which were presented to the students in the study material. Below are examples of students’ references to the De Morgan’s laws (left) and to the distributivity laws (right) found in the corpora (spelling and capitalisation preserved):

DeMorgan-Regel-1
DeMorgan-1
deMorgan-Regel-1
de-Morgan-Regel 1
De-Morgan-Regel-2
de morgan regel 2

Distributivitaet von Vereinigung ueber den Durchschnitt
Distributivität von Vereinigung über Durchschnitt
Distributivitaet von Durchschnitt ueber Vereinigung
DAS GESETZ DER DISTRIBUTIVITIT VON VEREINIGUNG UBER DURCHSCHNITT
der Distributivitaet 1

In the study material, the two De Morgan laws were labelled ‘‘De Morgan Regel 1’’ and ‘‘De Morgan Regel 2’’ and distributivity laws ‘‘Distributivität von Vereinigung über Durchschnitt’’ and ‘‘Distributivität von Durchschnitt über Vereinigung’’. As the examples illustrate, learners misspell the names and segment them in non-standard ways (hyphens in place of white-space, for instance), even those which were presented to them in a specific form.[36]

Moreover, a number of technical terms in mathematics (names of objects and relations) are multi-word units, for instance, ‘‘degrees of freedom’’ or ‘‘dot product’’. Much as in the case of named entities, different lexical variants denoting the same object may exist, for instance, ‘‘δ function’’, ‘‘Dirac’s delta function’’, and ‘‘Dirac’s delta’’ name the same concept. Multi-word constructions which incorporate symbolic expressions, such as ‘‘δ function’’ or ‘‘α-stable’’ (stochastic process), are not uncommon. Set theory itself has a few multi-word domain terms: ‘‘the universal set’’ (‘‘die Universelle Menge’’ in German) or ‘‘the powerset’’ (‘‘Potenzmenge’’, a compound in German).

Finally, certain conventional mathematical phrasings can be considered domain-specific collocations or *routine formulae* in the sense of Wray and Perkins (2000).[37] Examples include natural language translations of propositional connectives, such as ‘‘$\mathcal{A}$ if and only if $\mathcal{B}$,’’ ‘‘$\mathcal{A}$ and $\mathcal{B}$’’, ‘‘if $\mathcal{A}$, then $\mathcal{B}$’’, and other fixed phrases, such as ‘‘without loss of generality,’’ ‘‘what was to be shown,’’ or ‘‘This completes the proof.’’[38] (A full-text search for ‘‘This completes the proof’’ in the entire arXiv returned over 29000 hits.)[39] All of them have German, also multi-word, counterparts and occurred in our corpora.

[36] Of course, the examples can be recognised automatically using simple string matching.

[37] ‘‘[A] sequence, continuous or discontinuous, of words or other meaning elements, which is, or appears to be, prefabricated: that is, stored and retrieved whole from memory at the time of use, rather than being subject to generation analysis by the language grammar’’ (*ibid.*).

[38] A thematic list of most common English formulaic phrasings can be found in (Trzeciak, 1995).

[39] Full text search performed on `http://arxiv.org/find` on August 21, 2010.

Abbreviations Much like ordinary language, the language of mathematics uses abbreviations, i.e. shortened forms of words and phrases: initialisms, acronyms, or syllabic abbreviations. Aside from those found in ordinary language, e.g. ‘‘e.g.’’ or ‘‘i.e.’’ in English, mathematics uses its own domain-specific abbreviations: references to sides of mathematical formulas, ‘‘the left-hand side’’ and ‘‘the right-hand side’’, are often abbreviated with ‘‘l.h.s.’’ or ‘‘LHS’’ and ‘‘r.h.s.’’ or ‘‘RHS’’, the end of a proof is signalled with the Latin ‘‘q.e.d.’’ or ‘‘QED’’, a well-formed formula is a ‘‘wff’’, ‘‘if and only if’’ is shortened to ‘‘iff’’, etc. Some abbreviations are used in specific subareas of mathematics more often than in others: in probability theory, for instance, some of the standard terms are often abbreviated: ‘‘almost surely’’ with ‘‘a.s.’’, ‘‘infinitely often’’ with ‘‘i.o.’’, ‘‘almost every’’ or ‘‘almost everywhere’’ with ‘‘a.e.’’ Some abbreviations are so specific that without the knowledge of the particular field in which they are used, it is impossible to unfold them, for instance, the French-origin ‘‘càdlàg’’ or ‘‘cadlag’’ and its English equivalent, ‘‘RCLL’’. Examples of German abbreviations which occurred in the two corpora include different spelling variants of the following:

General language abbreviations:

d.h.	das heißt (*this means*)
bzw.	beziehungsweise (*respectively*)
Bsp.	Beispiel(e) (*example(s)*)
z.B.	zum Beispiel (*for example*)

Maths-specific abbreviations:

o.B.d.A.	ohne Beschränkung der Allgemeinheit (*without loss of generality*)
q.e.d.	quod erat demonstrandum
s.t.	such that

While most abbreviations are specific to the natural language of the discourse, Latin abbreviations, such as ‘‘q.e.d.’’, are used internationally. Interestingly, one of our students consistently used the English ‘‘s.t.’’ in the German discourse.

3.2.2.3 Syntactic phenomena

In general, the natural language part of the informal language of mathematics follows the syntax of the national language of the discourse, English, German, etc.[40] While in textbook and publication proofs most utterances

[40] (Up to certain irregularities discussed further in this section.)

(or sentences in this case) are in indicative mood, tutorial dialogue contains also other clause types (all examples from C-II):

Indicatives state unqualified mathematical facts,

Interrogatives ask questions, for instance, requesting a definition of a concept: ‘‘Was ist eine inverse Relation?’’ (*What is an inverse relation?*)

Imperatives command to perform actions, for instance, to state proof steps or give help: ‘‘Gib mir doch mal ein konkretes Bespiel wie man Beweise in der Mengenlehre loest!’’ (*Give me a concrete example of a proof in set theory!*) or ‘‘erklaere die Definition $R \circ S$ in Worten!’’ (*explain the definition of $R \circ S$ in words!*)

Exclamatives express emotions: ‘‘Schwachsinn!’’ (*Nonsense!*) or ‘‘Das beantwortet meine Frage nur zur Haelfte!’’ (*That's only half an answer to my question!*)

All the syntactic clause structures can be found in learner proofs in tutorial dialogue. The most frequent type of construction is the conditional. Zinn discusses conditionals in mathematics at length in his Chapter 4 (Zinn, 2004). We will not repeat the discussion on conditionals here nor in the section on semantics. Below, we only illustrate the complexity of the syntax of utterances involving conditionals found in the learner corpora, with three examples:

(15) wenn $A \subseteq K(B)$, dann $A \neq B$, weil $B \neq K(B)$
if $A \subseteq K(B)$, then $A \neq B$, because $B \neq K(B)$

(16) $\forall(x, y)$ gilt: wenn $(x, y) \in (R \circ S)^{-1}$ dann $(x, y) \in S^{-1} \circ R^{-1}$
und wenn $(x, y) \in S^{-1} \circ R^{-1}$ dann $(x, y) \in (R \circ S)^{1}$
$\forall(x, y)$ it holds: if $(x, y) \in (R \circ S)^{-1}$ then $(x, y) \in S^{-1} \circ R^{-1}$
and if $(x, y) \in S^{-1} \circ R^{-1}$ then $(x, y) \in (R \circ S)^{-1}$

(17) fuer $(a, b) \in (R \cup S) \circ T$ gilt: entweder $(a, x) \in R$ oder $(a, x) \in S$,
weil $(a, b) \in (R \cup S)$, wenn $(a, b) \in R$ oder $(a, b) \in S$
und gleichzeitig gilt $(x, b) \in T$
for $(a, b) \in (R \cup S) \circ T$ it holds: either $(a, x) \in R$ or $(a, x) \in S$
because $(a, b) \in (R \cup S)$ if $(a, b) \in R$ or $(a, b) \in S$
and at the same time $(x, b) \in T$ holds

The quoted utterances contain multiple clauses: subordinated or coordinated and subordinated. Their clause patterns can be summarised as:

wenn $\mathcal{A}$ dann $\mathcal{B}$ weil $\mathcal{C}$
wenn $\mathcal{A}$ dann $\mathcal{B}$ und wenn $\mathcal{C}$ dann $\mathcal{D}$
entweder $\mathcal{A}$ oder $\mathcal{B}$ weil $\mathcal{C}$ wenn $\mathcal{D}$ oder $\mathcal{E}$ und $\mathcal{F}$

Extended concatenation of clauses is unusual both in spoken and in written language. However, many occurrences of conjoined clauses of this kind can be found in our learner corpora. In terms of computational processing, this calls for a grammar formalism in which complex multi-clause utterances of this type could be modelled with sufficient generality. (In a context-free grammar, every instance of clause ordering would have to be modelled explicitly in order to obtain all the possible structural analyses; a suboptimal solution.) Specific to German is, moreover, the difference in word order between main clauses and subordinate clauses. The former exhibit the so-called verb-second word order (roughly speaking, the inflected verb is the second constituent), while the latter exhibit verb-last order (the inflected verb is the last constituent). The resulting dependencies require that the grammar formalism be expressive enough for the syntax–semantics interface to return valid interpretations.

Aside from clause structure complexity, informal mathematical language is also characterised by certain syntactic idiosyncrasies due to its mixed nature. Students' language in tutorial dialogue exhibits, additionally, syntactic irregularities which are normally never found in textbooks or scientific publications. These characteristics are illustrated in the following sections.

Syntactic categories of mathematical expressions In Section 3.2.2.1 (p. 113), we showed examples of mathematical expressions smoothly integrated into the syntax of natural language:

(18) $K(A \cup B)$ ist laut DeMorgan-1 $K(A) \cap K(B)$

(19) Wenn $x \in B$ dann $x \notin A$

(20) B enthaelt kein $x \in A$

(21) A auch $\subseteq B$

(22) $A \cap B$ ist $\in$ von $C \cup (A \cap B)$

In (18) and (19) symbolic expressions, terms and formulas, are used in place of complete valid constituents: subject and object noun phrases in (18) and main and dependent clauses in (19). This kind of symbolic expression embedding is easy to explain. The key observation here is that mathematical expressions can be naturally interpreted as corresponding to two linguistic syntactic types: clauses and noun phases, and the consistency in how mathematical expressions are embedded into natural language context stems from this correspondence. In most cases, mathematical formulas (proposition denoting) correspond to natural language clauses, while mathematical terms (object or type denoting) and *mentions* of mathematical formulas, as in ''$A \subseteq B$ is a formula'', correspond to noun phrases. This is in turn because in the symbolic language formula-forming operators correspond to natural language predicates (with ''be'' as a support verb if the operator does not have a verb reading), term-forming operators to natural language relational nouns, and atomic terms (variables and constants) to nouns.[41] (19) is a grammatical sentence under the standard grammar of German (and English) because the formulas' main operators fill in for the predicates (or their parts, as in the case of $\in$).

The next example, (20), illustrates another recurring type of embedding of symbolic expressions which on the surface have an appearance of formulas. The presence of a natural language sentential predicate signals the need for syntactic reinterpretation of the formula such that the utterance is paraphrased as ''B contains no x which is an element of A''. Under this interpretation, only the left-hand side of the formula is in the scope of the negation word preceding it, filling the role of a direct object of the main verb, ''contain'', pre-modified by the negation word. The remaining part of the expression serves as a post-modifying restrictive relative clause, of which the formula-forming operator is the main predicate (with ''be'' as a support verb). Thus, the syntactic chunk ''no $x \in A$'' is read as ''no x which is in A''.[42] Several observations can be made here: First, the interaction of symbolic expressions of type formula with the left linguistic context appears to be an artefact of formulas being written in *infix* notation. Thus, ''contains $x \in A$'' is licensed, whereas the same expression in prefix notation, ''contains $\in x\ A$'', would not result in a meaningful reading and it is questionable that a postfix notation, ''contains $x\ A \in$'', would read naturally. Second, the distribution of linguistic contexts which license such a reading is not random and includes categories which form valid constituents with individual-denoting (as opposed to eventuality-denoting) words in their right context: in English and German these are transitive verbs, nouns and adjectives, quantifiers, and negation words.[43] Finally, only individual-denoting constituents of a symbolic expression can interact with the preceding context. In order to recover the reading, a meaningful object-denoting substructure must be identified in the symbolic expression, based on its parse tree: the subexpression to the left of the main operator is the one which enters into a dependency relation with the left context, while the other substructure headed by the top-node becomes its dependent.

Finally, the last two examples show that mathematical expression ''fragments'' can be also embedded into natural language text. In (21) an adverb

[41] Formula mentions, such as the one presented, must be reinterpreted to be treated as a whole, a ''name'', in order to arrive at the right interpretation. The question of how to treat mathematical terms semantically – as definite descriptions, for instance – can be left aside at this point.

[42] Alternative readings could be ''no x such that it is in A'' or ''no such x that x is in A''.

[43] The list is based on an ad hoc analysis of textbook discourse. A further more systematic analysis of a large corpus of mathematical discourse is needed.

modifies a sentential predicate expressed in the symbolic language. (22) shows that formula-forming operators, which otherwise serve as predicates, can serve as names of objects formed by their predication. Here, the symbol $\in$ (''be an element of'') fills in for the nominal object of the predicate ''be''; similarly, $\subseteq$ could be used in place of the noun ''subset'' and $\cup$, an object-forming operator, would work in ''$A \cup B$ is a $\cup$ of A and B'' (a constructed example).

The latter two constructions illustrate a tendency towards *telegraphic style* in learner language in which symbolic notation is used as a kind of shorthand for the corresponding natural language wording. While the latter two forms are perhaps too informal to be encountered in textbooks, it is plausible that they can occur in written student homeworks or exams. In a computational processing framework this calls for a lexicon representation and an approach to parsing which would enable systematic treatment of symbolic expressions embedded within text, be it complete constituents or fragments, on a par with natural language lexemes and phrases.

Irregular syntactic constructions As a sublanguage, informal mathematical language admits of constructions which outside of mathematical discourse would be considered syntactically invalid. One type of syntactic irregularity is an artefact of how symbolic notation is verbalised (discussed in Section 3.2.1). For instance, an expression $A \cup B$, when spoken, will be typically read from left to right as it is written by substituting words for symbols: ''A union B'', resulting in a construction which is not only ungrammatical, but does not yield the intended semantics of ''the union of A and B'' under any standard interpretation of compounds of this type either.[44] Example (23) illustrates a similar construction in German which appeared in C-I:

(23) wenn A vereinigt C ein Durchschnitt von B vereinigt C ist, dann müssen alle A und B in C sein
If A union C is an intersection of B union C, then all A and B must be in C

Here, the student uses the construction ''NP vereinigt NP'' twice. This is a corrupt German participial construction with the verb ''vereinigen'' (*unify*) which in its grammatical predicate–argument structure requires a prepositional phrase ''mit + Dative'' (*with*). Another irregular syntactic construction resulting from writing an expression as it is spoken is illustrated below:

(24) Wenn (b, z) in R ist, ist dann a in R hoch minus eins?
If (b, z) is in R, then is a in R to minus one?

[44] The expression $A \cup B$ corresponds to a natural language construction involving two nouns, A and B, and a relational noun ''union (of)''. In an analogous construction in natural language, for instance ''friend of Peter and Paul'', the alteration ''Peter friend Paul'' is ungrammatical.

In this example, the student verbalises the notation of inverse relation as ‘‘hoch minus eins’’ (*to minus one*), the way it is normally read aloud when exponentiation is involved. The construction ‘‘hoch number’’ is syntactically marked: ‘‘hoch’’ as a modifier of a number category appears exclusively in the mathematical context, and normally only in spoken verbalisation.[45] The fact that it is found in type-written tutorial dialogue suggests that the learner adopted an informal conversational style of interaction and assumed that understanding spoken language style should be within the capabilities of the system's input interpretation component. Interestingly, non-canonical telegraphic syntax of this kind appears also in mathematical textbooks. Natho (2005, p. 109) quotes the construction ‘‘f injektiv’’ (*f injective*) with the copula verb omitted. This type of syntactic reduction is another manifestation of telegraphic style.

Syntactic ambiguities Finally, natural language structures, especially complex multi-clause utterances, are prone to syntactic ambiguities. A structural ambiguity introduced by the worded coordination is illustrated below:

(25) $x \in B$ und somit $x \subseteq K(B)$ und $x \subseteq K(A)$ wegen Voraussetzung
$x \in B$ and therefore $x \subseteq K(B)$ and $x \subseteq K(A)$ given the assumption

Exemplary alternative readings can be represented schematically as follows:

$$\begin{array}{llllllllllll} [\,[& \mathcal{A} & \text{und somit} & & \mathcal{B} &] & \text{und} & [& \mathcal{C} & & \text{wegen } \mathcal{D} &]\,] \\ [\,[& \mathcal{A} & \text{und somit} & [& \mathcal{B} & & \text{und} & & \mathcal{C} &]\,]\,[& \text{wegen } \mathcal{D} &]\,] \\ [& \mathcal{A} & \text{und somit} & [\,[& \mathcal{B} & & \text{und} & & \mathcal{C} &]\ [& \text{wegen } \mathcal{D} &]\,]\,] \end{array}$$

The previously presented examples (15) through (17) (p. 120) exhibit similar structural ambiguities. Since domain inference is needed to evaluate propositional content, a linguistic interpretation module alone cannot identify the most likely reading. However, its parser should be capable of parsing complex conjoined clauses of this type and identifying structurally ambiguous readings, be it by representing them in a compact way or enumerating alternative parses.

3.2.2.4 Semantic phenomena

Ordinary language and the language of mathematics sometimes use the same vocabulary, but its mathematical meaning differs from its meaning in natural language.[46] Quantifiers and connectives are examples of such words, often confused by learners. The natural language quantifier ‘‘any’’ can be used either in the existential (as in ‘‘Did you see any movie lately?’’) or universal

[45] The word ‘‘hoch’’ (*highly/upwards*) is an adverb in German and usually appears in participial constructions such as ‘‘hoch kompiliziert’’ (*highly complicated*).

[46] Examples of confusable vocabulary were shown when discussing the lexicon; see p. 115.

(as in, ‘‘Any dream will do.’’) sense. This ‘‘sloppiness’’ of natural language may lead to a confusion when ‘‘any’’ is used in a routine however imprecise mathematical construction ‘‘for any’’. A similar problem arises with ‘‘and’’ and ‘‘or’’. As a logical connective in mathematics ‘‘and’’ has a unique meaning: that of a truth functional conjunction. In natural language, however, ‘‘and’’ can have other meanings than that of a logical conjunction: for instance, that of a discourse marker introducing a rhetorical relation denoting result, implication, or temporal sequence, or that of an additive particle. In mathematics the meaning of ‘‘$\mathcal{A}$ or $\mathcal{B}$’’ can be paraphrased as ‘‘either $\mathcal{A}$ or $\mathcal{B}$ or both’’ and, naturally, different truth conditions apply to inclusive and exclusive disjunction. While natural languages typically do have a linguistic device to express the exclusive meaning (for instance, ‘‘either . . . or . . . ’’ in English) ‘‘or’’ may be used in both contexts. The following sections illustrate imprecision and other semantic phenomena in informal mathematical language which require special processing resources for computational interpretation.

Imprecision While mathematics is *the* precise discipline par excellence, its informal language is remarkably imprecise. Consider the following examples:

(26) B enthaelt kein $x \in A$
B contains no $x \in A$

(27) also gilt ferner, da A und B keine gemeinsamen Elemente haben, dass $K(A)$, definiert als $U \setminus A$, die Menge B enthält
therefore since A and B have no common elements, $K(A)$, defined as $U \setminus A$, contains the set B

(28) daraus folgt, dass $(z, y \in R^{-1}$ und (x, z) in S^{-1}
from that it follows that $(z, y \in R^{-1}$ and (x, z) in S^{-1}

(29) $(A \cap B)$ muss in $P((A \cup C) \cap (B \cup C))$ sein, da $(A \cap B) \in (A \cap B) \cup C$
$(A \cap B)$ must be in $P((A \cup C) \cap (B \cup C))$ since $(A \cap B) \in (A \cap B) \cup C$

In the first two utterances, the students used the predicate ‘‘enthalten’’ (*contain*); in (26), B, a first order set, is its subject and x, a set element, its object. In (27), $K(A)$, a first order set, is the subject and B, also a first order set, the object. The predicate ‘‘contain’’ is imprecise (ambiguous). In the context of set theory, containment may refer to subset/superset or element relation. In the context of symbolic mathematical expressions, containment could be also interpreted as structural composition; one expression being a structural subexpression of another.[47] In (26) the element relation is meant, while in (27) the subset relation is intended. Similarly ambiguous is the locative

[47] If in the previous context there would have been an assignment of B to a formula in which $x \in A$ is a subexpression, the structural composition reading would be plausible.

prepositional phrase with ''in'' in the next two examples. In (28) the element reading is intended. In (29), while the element reading more plausible, it is not clear whether the student realises the difference between the two relations considering the error in the dependent clause (A, B, and C are first order sets).

The examples illustrate the fact that in informal mathematical language mathematical concepts are named using common words which are imprecise (recall the examples from Table 3.4 on page 117) but which do have precise mathematical interpretations.[48] The same common word or construction may be used to name a class of conceptually related mathematical notions, especially if they are conceptualised as precisified subclasses of a more general concept, as is the case with different types of containment above.

In fact, in the course of learning mathematics, students are often explicitly told to *conceptualise* mathematical concepts as analogous to specific real-world images, that is, to build *conceptual metaphors* in their minds which visualise mathematical notions. Lakoff and Núñez (2000) take a radical stance on mathematical understanding in *Where mathematics comes from*, claiming that all of mathematics is a mental product which arises from our *embodied* minds, everyday experiences, and from human mind's unconscious *empirical* cognitive mechanisms, such as metaphors and image schemata. In line with Lakoff's prior cognitive linguistic theories, Lakoff and Núñez attribute (almost all) mathematical understanding to the process of understanding layers of *mathematical conceptual metaphors*, that is, inference-preserving mappings between conceptual domains: a source domain, from which metaphorical expressions are drawn, and a target domain, the domain which is being interpreted. Mathematical metaphors make it possible to understand complex, abstract mathematical notions (targets) in terms of simple, concrete notions from our everyday reality (source domains). For example, abstract sets can be understood via the (physical) *container* metaphor: The notion of a set is conceptualised as a container; a set is a container with things in it. The things may be simple things or sets of things. Given this image, we can conceptualise different configurations involving containers: one container inside another, as in the former examples, or two containers with different things in them:

(30) B vollstaendig ausserhalb von A liegen muss,
also im Komplement von A
B has to be entirely outside of A, therefore in the complement of A

(31) dann sind A und B vollkommen verschieden, haben keine gemeinsamen Elemente
then A and B are completely different, have no common elements

[48] Also Halmos (1970, p. 144) comments on natural language wording used for set relations.

‘‘Lying outside’’, (30), and ‘‘being different’’, (31), are informal natural language descriptions of an empty intersection of sets. The mental image of a container evokes a vague relation of similarity between containers (here, the property of two containers being different) and relations and properties associated with containers, such as location (here, of one container’s content).

Although the authors do not make specific claims as to the language phenomena resulting from the mapping, the theory explains the fact that the language used to talk about sets reflects the language used to talk about the source domain of the metaphor, containers: hence, we talk about sets ‘‘containing’’ elements, to express the set membership relation, and about sets ‘‘being contained in’’ or simply ‘‘being in’’ another set, to express the subset relation. The resulting ambiguity in the interpretation of the specific mathematical set relation meant is an artefact of the imprecision of the natural language phrasing. However, since the phenomenon is systematic, a computational interpretation component needs a representation of the imprecise concepts and an appropriate mapping to the possible specific mathematical interpretations. Notice moreover that this kind of ambiguity appears also in textbook discourse (recall, for instance, the previously quoted definition of set membership from (Bartle and Sherbert, 1982); see p. 96 of this chapter) which all the more motivates this as a basic requirement for a computational processing architecture. In our domain model specific mathematical relations are subsumed under more general relations reflecting the conceptual structure discussed above; see Section 6.2.1.

The metaphor mechanism can result in further imprecise wording: Following the container metaphor, students can of course talk about smaller and larger containers when referring to sets’ cardinalities:

(32) Der Schnitt von zwei Mengen ist kleiner gleich der kleineren dieser Mengen, also ist das Komplement des Schnitts größer gleich das Komplement der kleineren Menge

The intersection of two sets is smaller equal the smaller of these sets, so the complement of the intersection is larger equal the complement of the smaller set

Note that while natural language introduces imprecision, it is an imprecision in the sense of ambiguity, that is, a discrete set of possible interpretations (precisifications) exists. Mathematics is in general void of *vagueness* in that mathematical concepts are *precisely defined*. There exist, however, technical terms, also used in definitions, which are inherently vague. Consider, for instance, the mathematical uses of ‘‘almost all’’ (all except for finitely many or all except for a countable set) or ‘‘sufficiently large’’ (greater than some number).

Contextual operators Consider the following two examples from the corpora:

(33) Wenn alle A in $K(B)$ enthalten sind und dies auch umgekehrt gilt, muß es sich um zwei identische Mengen handeln
If all A are contained in $K(B)$ and this also holds the other way round, these must be identical sets

(34) S5: es gilt natürlich: $P(C \cup (A \cap B)) \subseteq P(C) \cup P(A \cap B)$
it holds of course: $P(C \cup (A \cap B)) \subseteq P(C) \cup P(A \cap B)$
S6: nein doch nicht... andersrum
no not that either... the other way round

''Umgekehrt'' and ''andersrum'' or their English counterpart, ''the other way round'', are complex operators which require contextual interpretation. In the first example, (33), ''the other other way round'' is ambiguous: the clause ''and this also holds the other way round'' may be interpreted as ''und alle $K(B)$ in A enthalten sind'' (*and all $K(B)$ are contained in A*) or as ''und alle B in $K(A)$ enthalten sind'' (*and all B are contained in $K(A)$*), the intended interpretation. Under the first interpretation, the entire dependent substructures of the head verb ''enthalten'', A and $K(B)$, are involved, whereas under the second, only parts of substructures, A and B, are involved (the directly dependent nodes, but not their dependents; assuming we analyse mathematical expressions in terms of dependency syntax as in natural language analysis). In (34) the entire dependent subtrees of the predicate expressed in the symbolic language, $\subseteq$, are involved, however, the scope of the semantic reconstruction involves content which appeared two dialogue turns prior to the turn with the operator; following S5 the tutor uttered ''Wirklich?'' (*Really?*) upon which the student revised his proof step in S6 with ''the other way round''.

''The other way round'' is a typical example of a *contextual operator*. Kay (1989) defines contextual operators as ''lexical items or grammatical constructions whose semantic value consists, at least in part, of instructions to find in, or impute to, the context a certain kind of information structure and to locate the information presented by the sentence within that information structure in a specified way''. Other items which have this property and which have been discussed in the linguistic literature include ''respective'', ''respectively'', and ''vice versa'' (Fraser, 1970; McCawley, 1970; Kay, 1989). Interpretation of operators of this type is non-trivial precisely due to their contextual and parasitic nature: the context needed for interpretation may span multiple clauses (or even dialogue turns in our case), it may contain multiple candidate arguments for the operator, and the candidates may appear in a variety of syntactic and semantic-dependency configurations. Computational interpretation must involve identifying the scope of the semantic reconstruction and a transformation process which recovers the implicit propositional content.

While the scope of ‘‘the other way round’’-like operators may span a number of clauses, the scope of ‘‘analogously’’, another contextual operator, may span entire larger discourses, as the following examples illustrate:

(35) S13: $(R \circ T)$ ist definiert als $\{(x, y)|\exists z(z \in M \wedge (x, y) \in R$
$\wedge (y, z) \in T)\}$
$(R \circ T)$ is defined as $\{(x, y)|\exists z(z \in M \wedge (x, y) \in R \wedge (y, z) \in T)\}$.
S14: $(S \circ T)$ ist genauso definiert.
$(S \circ T)$ is defined in the same way.
S15: $(S \circ T)$ ist analog definiert.
$(S \circ T)$ is defined in an analogous way.)

(36) Der Beweis von $(T^{-1} \circ S^{-1})^{-1} = (S \circ T)$ ist analog zum Beweis von $(T^{-1} \circ R^{-1})^{-1} = (R \circ T)$.
The proof of $(T^{-1} \circ S^{-1})^{-1} = (S \circ T)$ is analogous to the proof of $(T^{-1} \circ R^{-1})^{-1} = (R \circ T)$.

(37) Der Beweis geht genauso wie oben
The proof goes the same way as above

In (35) interpreting ‘‘analog’’ (*analogously*) requires an appropriate variable substitution in the definition of composition of relations which the student formulated two turns earlier. Note that the tutor did not accept the student’s phrasing with ‘‘genauso’’ (*the same way*) and asked for clarification: ‘‘Was heisst ‘genauso’?’’ (*What do you mean by ‘the same way’?*).[49] In (36), however, ‘‘analogously’’ is used in place of an entire proof which spanned about 15 student turns. In this case, the complete previous proof object would have to undergo a rewriting transformation involving multiple variable substitutions. In the case of definition, (35), the phrasing ‘‘genauso’’ was not accepted, however, following (37) the tutor accepted it in the case of a larger proof. This is justified because here ‘‘the same’’ is plausible to refer to the high-level proof structure, rather than the specific variable instantiations, as is the case of definition. ‘‘Proofs by analogy’’ of this type occur frequently in textbooks and publications.

From a computational point of view, interpreting ‘‘analogously’’ or ‘‘genauso’’ in the case of proof steps or entire proofs, would involve, first, identifying candidate objects in the previous discourse representation, which could undergo a transformation and, second, identifying parallels between the object currently under discussion and the candidate objects retrieved from the previous

[49] The tutor apparently overlooked a typo in the variable naming.

discourse. While in the case of ‘‘the other way round’’ the transformation is at the level of linguistic entities and can operate on linguistic representations, the transformation needed for ‘‘analogously’’ does not operate on linguistic entities, but rather on domain objects built up by a domain reasoner based on discourse analysis: a deduction system's proof or proof step representations, and is therefore outside of the scope of this thesis. Our approach to semantic reconstruction of ‘‘the other way round’’ will be presented in Chapter 6.

Adjectives Mathematical adjectives are interesting from the point of view of their semantic properties and their computational representation. Consider, for instance, the terms ‘‘left inverse’’ and ‘‘right inverse’’. In a set with a binary operation, $*$, and an identity element e, a is a left inverse and b is a right inverse if $a * b = e$. However, by convention, an element is called an ‘‘inverse’’ (or ‘‘two-sided inverse’’) when it is *both* a left inverse and a right inverse with respect to $*$. Thus, from the point of view a taxonomy of mathematical objects the *is-a* relation holds in a counter-intuitive direction: it is *not always* the case that a left inverse *is-a*n inverse and a right inverse *is-a*n inverse, which would be the case if prenominal modification worked the way it usually works with adjectives in natural language. The cases of ‘‘ideal’’ and ‘‘left/right ideal’’ are analogous in this sense. Typical attributive adjectives also exist in mathematics; ‘‘monotonic/monotone’’, as in ‘‘monotonic function’’, is an example.

The second class of interesting adjectives are those which can be used predicatively. Examples of such adjectives include properties of relations, such as symmetry, commutativity, etc. When expressed in an adjectival form they are part of copular constructions such as the one illustrated below:

(38) Da die Mengenvereinigung kommutativ ist, . . .
Since set union is commutative, . . .

When formalised mathematically, commutativity of a binary operation $*$ on a set is defined as $x * y = y * x$ for all set elements x and y; for set union this would be instantiated as $A \cup B = B \cup A$, where A, B are sets. In this representation, a functional operator is involved and a structural result is defined. In natural language, as in (38), commutativity is predicated of set union. Informally, this could be represented symbolically as $\texttt{Commutative}(\cup)$, that is, a property is predicated of a function. Thus, the structure of the two representations is different and needs to be mapped. The same holds of other relation and function properties such as ‘‘symmetric’’, ‘‘distributive’’, ‘‘connected’’, etc. In general, the meaning of mathematical adjectives, denoting properties of mathematical objects, is formally defined. A language understanding component needs to be able to represent a mapping between the natural language adjectival use and the formal representation. In particular, in a tutorial dialogue system, this mapping has to link to an automated deduction system's internal representation, so that the validity of an assertion such as (38) can be verified.

Verbs In the course of problem solving learners verbalise ‘‘actions’’ which they intend to perform on terms and formulas before they actually carry out the formal operation. The following examples illustrate this:

(39) Ich zerlege jetzt die Potenzmenge:$P(C\cup(A\cap B)) \supseteq P(C)\cup P(A\cap B)$
I'm now splitting the powerset: $P(C \cup (A \cap B)) \supseteq P(C) \cup P(A \cap B)$

(40) Ich schätze die Vereinigung der Teilmenge ab $P(C) \cup P(A \cap B) \supseteq P(A \cap B) \supseteq A \cap B$
I'm estimating the union of the subset $P(C) \cup P(A \cap B) \supseteq P(A \cap B) \supseteq A \cap B$

(41) Nun wendet man das Relationenprodukt nochmals an, oder?
The relation product should be applied now, right?

(42) damit kann ich den oberen Ausdruck wie folgt schreiben:
$K((A \cup B) \cap (C \cup D)) = K(A \cup B) \cup K(C \cup D)$
thus I can write the above expression as follows:
$K((A \cup B) \cap (C \cup D)) = K(A \cup B) \cup K(C \cup D)$

This kind of language is characteristic of Tall's procept world (see Section 3.1.2 (p. 88)) in which focus is on actions, procedures, and algorithms. In order to obtain a complete interpretation of the intended proof step a formalisation of meanings of such ‘‘actions’’ would be needed.

The information about the fact that elements of the procept language occurred in a student's solution could be useful for the tutoring system's pedagogical module to reason about the student's knowledge state. This, however, means that an automated system would have to be able to verify whether the result of the operation actually performed on a symbolic expression can be indeed considered an instance of ‘‘splitting’’, ‘‘estimating’’, ‘‘applying’’, or ‘‘(re-)writing’’. This would in turn mean that the semantics of these actions would have to be operationalised. While ‘‘applying’’ a lemma or a theorem or ‘‘rewriting’’ an expression could be formalised in relatively straightforward way[50] a symbolic operationalisation of ‘‘splitting’’ is not so obvious; notice moreover that in the quoted example (39) the argument of the verb ‘‘split’’ has to be type recast: it is not the powerset object that is being ‘‘split’’, but rather

[50] (for instance, as a two-place function which takes arguments of types math expression and theorem and returns a result of type math expression which should have the property that it can be derived from the original math expression in one step using theorem)

the term headed by the powerset operator. Further similar examples will be discussed in the next section when we talk about bridging references.

3.2.2.5 Discourse phenomena

The discussion of discourse phenomena in mathematical discourse should perhaps start with an introduction on denoting. Mathematics is a tricky area in this respect; we will not attempt even a brief digression into the philosophical – ontological or epistemic – aspects of mathematics as these areas are outside of the scope of this work. The purpose of this section is far more down-to-earth: in the following sections, we will merely illustrate a number of discourse reference phenomena in proofs. In relation to referring, two points need to be mentioned about the universe of discourse.

Mathematics is about mathematical objects and, even more importantly, relations between them. At the conceptual level, mathematical discourse talks about *mathematical entities*, makes statements, *propositions* or *claims*, about these entities and ascribes *mathematical properties* to both the entities and the propositions. Mathematical objects – non-physical, timeless and spaceless, formally defined abstract entities – are evoked in mathematical discourse by their names. The words that name them are technical terms of mathematics. Mathematical objects in the domains of our corpora include sets, relations, and operations on sets and relations (set union, intersection, relation composition, etc.) which are themselves mathematical objects too.

Although in principle all of mathematics can be done in the mind and mathematical concepts can be considered purely mental constructs which do not need words, mathematics is of course communicated: in natural language, as in our experiments, or using other means, such as diagrams or graphs. Words, phrases, and sentences of the formal mathematical language, *mathematical expressions*, are symbolic textual representations of mathematical objects, relations, and propositions. This structured textual notation can be written in a precise formal way (as is the case in formal logic or proof theory) or semi-formally. We already discussed properties of the symbolic language in Section 3.2.1. The written representations are of course themselves mathematical objects and mathematical discourse talks about them as well. Thus, among reference phenomena, aside from the usual anaphoric references, other types of references are to be expected in mathematical discourse: references to the textual mathematical signs (notation) or their parts and references to mathematical propositions or sets of propositions which form a proof or part of a proof, that is, larger mathematical discourse objects. We discuss and illustrate these phenomena in the following sections.

Referring to domain objects Both definite and bare noun phrases can be used as specific references to refer to domain objects or as generic references to refer to domain concepts. For instance, ‘‘die Vereinigung’’ (*the union*) in (43) is a specific reference, whereas ‘‘die Potenzmenge’’ (*the powerset*) in (44) is a generic reference to powerset as a type:

(43) Die Vereinigung der Mengen R und S enthaelt alle Element aus R und alle Element aus S.
The union of the sets R and S contains all elements from R and all elements from S

(44) und für die Potenzmenge gilt: $P(C \cup (A \cap B)) = P(C) \cup P(A \cap B)$
and for the powerset it holds: $P(C \cup (A \cap B)) = P(C) \cup P(A \cap B)$

The interpretation of the reference ‘‘Potenzmenge’’ in (45) below is unclear:

(45) S1: $A \subseteq (A \cup C)$, $B \subseteq (B \cup C)$, also $(A \cap B) \subseteq ((A \cup C) \cap (B \cup C))$
$A \subseteq (A \cup C)$, $B \subseteq (B \cup C)$, thus $(A \cap B) \subseteq ((A \cup C) \cap (B \cup C))$
S2: Potenzmenge enthaelt alle Teilmengen, also auch $(A \cap B)$
Powerset contains all subsets, thus also $(A \cap B)$

S2 in (45) can be interpreted as an informal paraphrase of the definition of a powerset, in which case the reference is generic, or the learner may have meant the powerset of the specific instance of a set in S1, $((A \cup C) \cap (B \cup C))$, in which case the reference is specific.

Aside from evoking defined objects, mathematical discourse may contain references to named theorems, lemmata, definitions, or proofs. These are also mathematical objects and are often referred to by their proper names as in (46):

(46) Ich benutze das Extensionalitaetsprinzip
I’m using the Extensionality Axiom

The definite noun phrase ‘‘das Extensionalitaetsprinzip’’ (*Axiom of Extensionality*) is a non-anaphoric reference to a class of statements intentionally equivalent to the following:

$$A = B \Leftrightarrow \forall\, x\ (x \in A \Leftrightarrow x \in B), \text{ where } A, B : \text{sets}$$

Other examples of named mathematical objects of this type in our domains include: ‘‘De Morgan Regeln’’ (*De Morgan Laws*) or ‘‘Distributivgesetz’’ (*Distributive property*). Proof methods or strategies, likewise, have names, for instance, ‘‘indirect proof’’ or ‘‘proof by contradiction’’, ‘‘(Cantor’s) diagonal proof’’; specific proofs can be named entities as well, for instance, ‘‘the Euclid’s proof’’ (of the Pythagorean theorem), ‘‘the Wiles’ proof’’, or ‘‘the Hales proof’’. In most contexts, occurrences of these references are non-anaphoric.

Referring to (parts of) symbolic notation When mathematics is committed to written form, referring devices can be also used to relate to symbolic expressions in discourse or to their parts. Both direct – anaphoric – and indirect – bridging – references to (parts of) symbolic notation can be found in mathematical discourse. Both types of references are illustrated below.

Direct reference In direct reference a *coreference relation* exists between two discourse referents: the one introduced by a referring expression (*anaphor*) and another one introduced previously (*antecedent*); the two expressions denote the same entity. Prototypical anaphoric references are pronouns:[51]

(47) Da, wenn $A \subseteq K(B_i)$ sein soll, A Element von $K(B_i)$ sein muss. Und wenn $B_i \subseteq K(A)$ sein soll, muss es$_i$ auch Element von $K(A)$ sein.
Because if it should hold that $A \subseteq K(B)$, A must be an element of $K(B)$. And if it should hold that $B \subseteq K(A)$, it must be an element of $K(A)$ as well.

(48) S1: Wie ist $R \circ S$ definiert?
How is $R \circ S$ defined?

T1: $R \circ S := \{\ (x, y) \mid \exists z_i(z_i \in M \wedge (x, z_i) \in R \wedge (z_i, y) \in S\}$
S4: ist z_i nur fuer die Definition eingefuehrt oder hat es$_i$ einen anderen Sinn?
is z introduced only for the definition or does it have a different meaning?

In (47), the pronoun ''es'' (*it*) refers to a term in a formula, a set variable B in the previous clause. The syntactic function of the anaphor, subject of the clause, is parallel to the syntactic function of the antecedent in the formula verbalisation. Syntactic parallelism between the anaphor and a candidate antecedent is used in computational anaphor resolution as a strong indicator of coreference. Similarly, in (48) the pronoun ''es'' is referring to a variable naming a member of a set, x, which was first introduced earlier in the dialogue.

Coreference between variables in mathematics depends on the type of denotation the given variable has (specific unknown vs. continuous unknown vs. arbitrary fixed object, and so on), the logical structure of the argument (function and scope of the discourse segment in which the variable occurs), and quantification (the same variables in two existentially quantified formulas do not necessarily corefer).[52] The very notion of a variable, the meaning of variables, and quantification have been shown to cause learners major difficulties (Epp, 1999; Dubinsky and Yiparaki, 2000; Selden and Selden, 2003). A typical error in the use of variables from one of our corpora is shown in the examples that follow.

[51] Coreferring discourse entities are marked with matching subscripts.

[52] See (Kapitan, 2002) for a discussion on the nature of variables in mathematics.

(49) S18: Daraus folgt $(R \cup S) \circ T = \{(x_?, y) \mid \exists z(z \in M \wedge (x, z) \in \{x_? \mid x_? \in R \vee x_? \in S\} \wedge (z, y) \in T)\}$

From that follows $(R \cup S) \circ T = \ldots$

T19: Was bedeutet die Variable$_i$ x_i bei Ihnen?

What is the meaning of the variable x?

S19: x_i hat zwei Bedeutungen es$_i$ kommt in zwei verschiedenen Mengen vor

x has two meanings it appears in two different sets

T20: Benutzen Sie bitte fuer die zwei verschiedenen Bedeutungen von x zwei verschiedene Bezeichnungen.

Please use two different designations for the two different meanings of x.

In (49) the same name, x, is introduced to denote different entities which are moreover of different types: a variable in a pair and a set member variable in a set constructor. This kind of ambiguous designation is infelicitous a proof, so the tutor asks for clarification, ‘‘Was bedeutet die Variable x bei Ihnen?’’ (‘‘die Variable$_i$ x_i’’ is an example of appositional anaphoric reference). An anaphor appears also in the clarification subdialogue: the pronoun ‘‘es’’ in the second clause of S19 corefers with x in the preceding clause and in the tutor’s turn, however, a coreference chain cannot be established with the previous occurrences of x due to the ambiguous designation.

In the last examples pronominal adverbs refer to terms and formulas, (50) and (51), and an anaphoric epithet identifies an expression by its type, (52):

(50) S1: $[\ R \circ S\ ]_i := \{(x, y) \mid \exists z(z \in M \wedge (x, z) \in R \wedge (z, y) \in S)\}$

S2: Nun will ich das Inverse [davon]$_i$

Now I want the inverse of that

(51) Dann gilt fuer die linke Seite, wenn $[\ C \cup (A \cap B)\ ]_i = [\ (A \cup C) \cap (B \cup C)\ ]$ der Begriff $A \cap B$ dann ja schon dadrin und ist somit auch Element [davon]$_i$.

Then for the left side, if $C \cup (A \cap B) = (A \cup C) \cap (B \cup C)$ the term $A \cap B$ is already there and thus also an element of it

(52) T: $[\ R \circ S\ :=\ \{\ (x, y) \mid \exists z(z \in M \wedge (x, z) \in R \wedge (z, y) \in S)\}\ ]_i$.

S: So, und was ist das M in [der Formel]$_i$?

Right, and what is the M in the formula?

Other examples of anaphoric epithets include ‘‘the term’’, ‘‘the variable’’, ‘‘the constant’’, named results of operations (‘‘the sum’’, ‘‘the union’’, ‘‘the factors’’), named parts of symbolic expressions (‘‘the denominator’’), etc.

Indirect reference *Bridging* is a term introduced by Clark (1975) for definite noun phrases identifying a referent which has not been introduced explicitly, but which is ''associated'' with a previously evoked entity.[53] Bridging references can be used to identify mathematical expressions by their typographical features or physical properties (''the left side''), the linear order of their constituents (''the first term''), their structural groupings or delimited subexpressions (''the bracket''), or the type of object they denote (''the complement'', when it refers to a term headed by the complement operator). The following dialogue fragment exemplifies the phenomenon:

(53) T1: Bitte zeigen Sie: $A \cap B \in P((A \cup C) \cap (B \cup C))$!
Please show: $A \cap B \in P((A \cup C) \cap (B \cup C))$!
S1: Distributivitaet von Vereinigung ueber Durchschnitt:
$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ Hier dann also:
$C \cup (A \cap B) = (A \cup C) \cap (B \cup C)$ Dies fuer [die innere Klammer $]_i$. Auf [der linken Seite $]_j$ $A \cap B$. Hierfuer gilt Fall 10: Falls $A \in P((A \cup C) \cap (B \cup C))$ und $B \in P((A \cup C) \cap (B \cup C)) = A \cap B \in P((A \cup C) \cap (B \cup C))$
Distributivity of union over intersection: . . . So here: . . . This for the inner bracket. On the left side $A \cap B$. Case 10 applies here: If . . . and . . .
S2: Dann gilt fuer [die linke Seite $]_j$, wenn $C \cup (A \cap B) = (A \cup C) \cap (B \cup C)$ der Begriff $A \cap B$ dann ja schon dadrin und ist somit auch Element davon.
Then for the left side it holds, if . . . the term $A \cap B$ is already there and thus also an element of it
S3: $A \cap B$ auf [der linken Seite $]_j$ ist $\in$ von $C \cup (A \cap B)$, was ja nur durch C erweitert wird. Es kommt auf [der rechten Seite $]_k$ ja nur C als Vereinigungsmenge zu $A \cap B$ hinzu.
$A \cap B$ on the left side is $\in$ of $C \cup (A \cap B)$, which is extended only by C. On the right side is only C intersected with $A \cap B$.

The definite noun phrases ''die innere Klammer'' (*the inner bracket*), ''die linke Seite'' (*the left side*) and ''the right side'' refer to structural parts of the formula in T1 and they are all used in a bridging sense: ''the left side'' and ''the right side'' refer to the terms left and right of the top-node operator in the formula (rather than to the general areas to the left and right, respectively) while ''the inner bracket'' refers to a bracketed subterm embedded in another bracketed term, rather than to a bracket itself in the sense of a grouping element. (In English, of course, yet another interpretation of the reference ''bracket'', without the adjectival modification, would be possible in algebra. Lexical interpretation is, as always, dependent on the domain; here, mathematical

[53] Other terms used for this kind of reference are ''indirect anaphora'' (Chafe, 1972, 1976), ''associative anaphora'' (Hawkins, 1978), or ''inferrable'' (Prince, 1981).

subarea). The reference ‘‘die innere Klammer’’ is in this case unfortunately ambiguous: the singular ‘‘Klammer’’ may refer to either $(A \cup C)$ or $(B \cup C)$ both of which are bracketed subterms of the term $P((A \cup C) \cap (B \cup C))$; the plural ‘‘Klammern’’ was most likely intended, but mistyped.

The next set of examples, (54) through (56), illustrate bridging references to terms by means of the names of objects which the terms denote:

(54) T1: Bitte zeigen Sie: [$K((A \cup B) \cap (C \cup D))$]$_?$ $=$ ([$K(A)$]$_?$
$\cap$ [$K(B)$]$_?$) $\cup$ ([$K(C)$]$_?$ $\cap$ [$K(D)$]$_?$)!
Please show: ...!

S2: de morgan regel 2 auf [beide komplemente]$_i$ angewendet
de morgan rule 2 applied to both complements

(55) S2: hab mich verschrieben [$P((A \cup C) \cap (B \cup C))$]$_?$
$=$ [$P(C \cup (A \cap B)$]$_?$
made a typo $P((A \cup C) \cap (B \cup C)) = P(C \cup (A \cap B)$

S5: habe probleme mit [der potenzmenge]$_i$, kann sie$_i$ nicht
ausrechnen bzw mir sie$_i$ vor augen fuehren!
have problems with the powerset, can’t calculate it, can’t see it

(56) S33: Nach Aufgabe W ist $(S \circ (S \cup R)^{-1})^{-1} =$ [$((S \cup R)^{-1})^{-1}$
$\circ S^{-1}$]$_i$
By Exercise W: ... holds

S34: Dies$_i$ ist nach Theorem 1 gleich [$(S \cup R) \circ S^{-1}$]$_j$
This is by Theorem 1 equal to $(S \cup R) \circ S^{-1}$

S35: Ein Element (a, b) ist genau dann in [dieser Menge]$_j$,
wenn es ein $z \in M$ gibt mit $(a, z) \in S \cup R$ und $(z, b) \in S^{-1}$
An element (a, b) is in this set if and only if there is an $x \in M$
such that $(a, z) \in S \cup R$ und $(z, b) \in S^{-1}$

The quantified noun phrase ‘‘beide Komplemente’’ (*both complements*) in S2 of (54) refers to a pair of terms headed by the complement operator in T1. The plural in this case is multiply ambiguous. First, there is an ambiguity between the distributive and collective reading, and second, there are five complement-headed terms in the preceding formula. It is clear, however, that only two pairs of those are equally plausible as antecedents: $K(A)$ and $K(B)$ or $K(C)$ and $K(D)$; in fact, De Morgan rule has to be applied to both, pairwise.

There are two ways of interpreting the definite noun phrase ‘‘der Potenzmenge’’ (*the powerset*$_{Dat.}$) in S5 of (55). On the one hand, it may be referring to a term headed by the powerset operator in S2 (rather than the powerset operator itself) which contains the following expression: $P((A \cup C) \cap (B \cup C)) = P(C \cup (A \cap B))$. Under this interpretation, the reference is ambiguous since there are two powerset-headed subexpressions. On the other hand, it is more plausible to interpret it non-anaphorically, as a generic reference. Since the student had a general problem in understanding the concept of a powerset, it is unclear which one he meant.

In (56) the definite noun phrase ‘‘diese Menge’’ (*this set*) in S35 is again a bridging reference to the set defined by the composed relation denoted by $(S \cup R) \circ S^{-1}$ in S34. Yet another related type of bridging reference, of which we did not have examples in our corpora, are references to structures by means of their underlying objects; in the context of groups, for instance, given a set G and a binary operation $*$, one could refer to ‘‘the group G’’. Bridging references of this kind occur frequently in textbook discourse.[54] (56) also exemplifies a discourse deictic reference to a part of a mathematical expression: ‘‘dies’’ (*this*) in S34 points at the term on the right-hand side of the equality in S33.

Ganesalingam suggests that Zinn's analysis of structured mathematical terms which makes their subterms available for reference is incorrect: ‘‘[Zinn's analysis] frequently makes incorrect predictions about anaphor, even though this is one of the great strengths of Discourse Representation Theory. For example, consider the discourse: ‘2 + 15 is prime. It is divisible by 1 and 17 (only).’ Zinn's analysis incorrectly predicts that ‘2’ is an available anaphoric antecedent at the end of this discourse (Zinn, 2004, pages 106–7)’’ (Ganesalingam, 2009, p. 20). Considering the phenomena illustrated above, Zinn's analysis appears well-justified; even the quoted example could continue along the lines of ‘‘The left term is prime’’, for which, clearly, ‘2’ would need to be an available antecedent. In fact, Zinn's example (93a) on the quoted page 106: ‘‘$1, 1, 2, 3, 5, 8, 13, 21, ...$ in which [the first two terms]...’’ also supports this, as do his other examples (43c–e) on page 74 which illustrate the same phenomenon (albeit under an unfortunate heading of ‘‘Deictic form’’).

The question of which substructures of mathematical expressions should be available for reference does not have an obvious answer. In Section 3.2.1.2 we tried to show that certain substructures of mathematical expressions can be considered salient: they are valid constituents, in terms of the expression's tree structure, and they are distinct in the Western-tradition infix notation. Constituent structure analysis is also supported by studies on human perception of mathematical expressions (Jansen et al., 1999, 2000, 2003). These studies and the observations on referring from our corpora suggest that both atomic and complex subterms (along with information on their bracketing) should be available for reference. Now, the operator nodes of the expressions would need to be modelled too if meta-level discussion on mathematical expressions

[54] Wells (2003, p. 239) points out that this is an example of parameter suppression.

were to be allowed (a student could refer to ‘‘the plus sign’’ for instance), as well as the type of their result (see examples (54) and (56)). That is, not only ‘‘K’’ and ‘‘$\circ$’’ as the symbols themselves can be candidate antecedents, but the expressions headed by the operators need to be available, as already mentioned, along with the information on the type of objects they denote (here: a set; the type of the result of the complement operation and of relation composition).

Referring to propositions Both in our data as well as in narrative mathematical discourse pronouns, demonstratives, and adverbial pronouns are used to refer to propositions as well as sequences of propositions which form a proof. The examples below illustrate this:

(57) S11: $\exists z \in M$, so dass $(x, z) \in S^{-1}$ und $(z, y) \in R^{-1}$
$\exists z \in M$ such that $(x, z) \in S^{-1}$ and $(z, y) \in R^{-1}$
T18: Richtig. Wissen Sie, ob ein solches z existiert?
Correct. Do you know whether such z exists?
S12: Nein
No
T19: Erinnern Sie sich daran, dass [es ein z gibt mit $(x, z) \in S^{-1}$ und $(z, y) \in R^{-1}$]$_i$.
Do you remember that there is a z such that $(x, z) \in S^{-1}$ and $(z, y) \in R^{-1}$.
S13: Ja, ich habe es$_i$ vorausgesetzt
Yes, that was the assumption

(58) S7: Also ist [$(z, x) \in S$ und $(y, z) \in$ R]$_i$ und damit$_i$ auch [$(y, x) \in R \circ S$]$_j$
Therefore $(z, x) \in S$ and $(y, z) \in R$ holds and by that also $(y, x) \in R \circ S$
S8: [Somit]$_j$ ist $(x, y) \in (R \circ S)^{-1}$
Given that it holds that $(x, y) \in (R \circ S)^{-1}$

In (57), the pronoun ‘‘es’’ (*it*) is used (S13), as in ordinary discourse, to refer to a proposition, in this case, an assumption restated in the tutor’s turn (T19). More interesting are references using adverbial pronouns exemplified in (58). ‘‘Damit’’ (*with this*) in S7 refers to the proposition stated in the first conjunct of the coordinated clauses. ‘‘Somit’’ (*with that*) in S8 may refer to the conjunction of the assertions in S7 or only to the last assertion (marked with *j* in the example). On the one hand, in most cases, as here, references of this kind are underspecified in terms of their scope. On the other hand, their function is to signal the logical structure of the argument: the antecedent of ‘‘somit’’ or ‘‘damit’’ provides justification for the subsequent statement. In order to resolve the scope of such references, and so to reconstruct the intended logical structure of the proof, domain reasoning is needed.

Table 3.5: Categories of solution-related student contributions

Category	Description
Proof contributions	
Proof step	Contributes a proof step or part of a proof step
Proof strategy	States a solution strategy to be adopted
Proof structure	Signals solution structure
Proof status	Signals the status of the (partial) solution
Meta-level	
Self-evaluation	States an evaluation of own step
Restart	Signals that a new attempt at a proof is being started
Give up	Signals abandoning the solving task

Signalling proof structure and status Proofs are structured discourses. The discourse structure and linguistic realisation of a proof are dictated by the employed reasoning: the proof method and the sequence of inferences. Certain proof types have a characteristic form and elements: a proof by induction includes a base step part and an inductive step, a proof by contradiction starts with the assumption of the negated proposition and ends with a contradiction, and proof by cases comprises a sequence of case distinctions. The logical structure of the reasoning is made explicit in a proof using linguistic means: there exists conventional wording typically used to signal the proof method employed, the proof step elements (assertions, justifications, etc.), and the end of the proof. Aside from these proof components, students' proofs constructed in an interactive setting contain contributions which are typically not found in textbooks nor scientific publications. Table 3.5 shows a classification of student contributions which add information about the solution being constructed, that is, contain information related to the proof.[55]

From the point of view of their function, solution-related contributions can be divided into object-level and meta-level types. At the object-level, that is, at the level of the actual proof, we found four categories of contributions in the corpora: *Proof steps* are the actual complete or partial proposed steps in a proof. A minimal proof step consists of a proposition. The proposition may be an inferred assertion or an assumption. A complete inferred proof step consists of an assertion and a justification (a warrant) of the validity of the inference (by reference to proved claims or axioms and valid inference rules). The assertion can be formulated as a formal statement or a natural language statement in an indicative or conditional/hypothetical mood. A justification of a claim can be signalled using discourse connectives (in German: ‘‘aber’’, ‘‘und’’, ‘‘weil’’, ‘‘da’’, ‘‘dann’’, etc.; in English: ‘‘thus’’, ‘‘hence’’, ‘‘therefore’’, ‘‘because’’, etc.), other adverbial connectives,

[55] A broader characterisation of utterance types in our corpora will be presented in Chapter 4.

such as those discussed in the previous section (‘damit’’, ‘‘somit’’, ‘‘deshalb’’, ‘‘also’’), or descriptively using appropriate wording, for instance, ‘‘aufgrund des Extensionalitaetsprinzips’’, ‘‘aus Symmetriegründen’’ (*Due to extensionality/symmetry*), or ‘‘Begründung: . . . ’’ (*Justification: . . .*) Much like the adverbial pronouns, discourse connectives are scope bearing, but their scope is in many cases underspecified.[56] In most cases, moreover, the link between a new proposition and the previous propositions is not overtly given at all. Note that underspecification manifested in unclear scope of discourse markers signalling the logical structure in proofs is present also in textbooks. Again, in order to resolve the underspecified scope, *human-level* deductive reasoning is needed, that is, knowledge beyond mere semantic interpretation.

A declaration of *proof strategy* is a statement which does not bring the proof forward, but based on which the intended line of reasoning to follow can be anticipated. It can be signalled using wording such as ‘‘Beweis durch $\subseteq$ und $\supseteq$’’ (*Proof by $\subseteq$ and $\supseteq$*) or ‘‘es genügt zu zeigen...’’ (*it is enough to show...*), etc. By *proof structure* we mean explicit signals of a proof’s structural composition. This includes utterances such as ‘‘Schritt 1:’’ (*Step 1*) or ‘‘Ich mache eine Fallunterscheidung’’ (*I’m making a case distinction*). *Proof status* is a category for utterances which signal the current state or status of the proof, for instance, ‘‘q.e.d.’’, ‘‘Damit ist insgesamt gezeigt...’’ (*With that we have shown...*), or a more informal ‘‘Hälfte geschafft’’ (*Half done*).

Unlike proofs in textbooks or scientific publications, students’ solutions may be invalid (false) or not goal-oriented; a student may be going in the wrong direction or may not know at all how to proceed. In proofs constructed with tutor’s assistance, students can communicate this kind of meta-level information about their solution to the tutor. While all the proof contribution categories are also found in scientific publications, the latter contribution types are more likely to appear only in pedagogical contexts. Among meta-level solution-related communication, three types of contributions were found in the corpora: *Self-evaluations* are student’s own evaluations of the validity, granularity, or relevance of a proof step (or steps) which he proposed. Examples of such utterances include: ‘‘ich habe die falsche Richtung benutzt’’ (*I used the wrong direction (of an implication)*) or ‘‘Korrektur:...’’ (*Correction:...*); the latter being an implicit self-evaluation. If a solution attempt is not successful, a student can *restart* and try a new solution signalling that the previous one is abandoned: ‘‘Ich beginne den Beweis neu’’ (*I’m starting the proof anew*)

[56] Adverbs such as those mentioned here take two arguments, both of which may span multiple assertions. In English, one argument immediately follows and the other may take scope over just the previous assertion (here: a previous step) or over a larger discourse (here: a number of proof steps, along with their justifications; a subproof).

or ''Wieder von vorne'' (*Once again from the beginning*). Finally, if a student cannot find a solution, he may decide to give up: ''Ich gebe auf'' (*I'm giving up*), ''Bitte die richtige Antwort!'' (*Show me the right solution, please!*).

3.3 Pragmatic aspects of mathematical discourse

From a pragmatic[57] point of view the main purpose of the language of mathematics is to convey ''mathematical content'', that is, factual propositional information about mathematical objects, relations, and properties. Thus, on the one hand, in Brown and Yule's terminology mathematics is a *transactional discourse* (Brown and Yule, 1983). On the one hand, a mathematical proof is a form of *persuasive discourse*, a ''validating act'', in which the speaker (the proof's author) is attempting to convince the hearer/reader that certain mathematical facts hold (Hersh, 1993). A proved mathematical assertion becomes a theorem and can be invoked in another proof to make new inferences. Assertions without proofs can only appear if they are postulated to be true (axioms), locally assumed to be true (hypotheses), or explicitly declared as such (conjectures). From a pedagogical point of view a proof is also an educational tool: by constructing a proof a learner is attempting to convince, himself and the teacher that his argumentation is based on understanding, rather than on mere repetition of memorised theorems and lemmata, and he is discovering relations between mathematical concepts, thereby deepening his understanding (Hanna, 1990; Sfard, 2001); hence the importance of the learner showing (justifying) how the proposed proof steps have been derived. Much like in any other dialogue situation, participants of mathematical dialogue follow certain *cooperative principles*[58] and make assumptions as to the stock of knowledge that is shared between them. On the part of the tutor, cooperativity involves contextual interpretation: resolving underspecified scopes, covert arguments, and references, both in the natural language and in the symbolic notation (discussed in Section 3.2.1.3) as well as resolving semantic ambiguities which are due to imprecise language (Section 3.2.2.4). At the proof-level,

[57] (in a technical sense of the word)

[58] Grice's Cooperative Principle (Grice, 1975) states that a conversational contribution should be made ''such as is required, at the stage at which it occurs, by the accepted purpose or direction of the talk exchange.'' Cooperative communication is governed by conversational maxims: *Quality:* Try to make your contribution one that is true. 1. Do not say what you believe to be false. 2. Do not say that for which you lack evidence; *Quantity:* 1. Make your contribution as informative as is required (for the current purposes of the exchange). 2. Do not make your contribution more informative than is required; *Relation:* Be relevant; *Manner:* Be perspicuous. 1. Avoid obscurity of expression. 2. Avoid ambiguity. 3. Be brief. Avoid unnecessary prolixity. 4. Be orderly.

it involves filling in the gaps in coarse-grained reasoning. These concern also mathematical prose. From the pedagogical point of view, cooperativity may also involve ignoring certain low-level errors in favour of the higher goal of teaching mathematical argumentation (Section 3.2.1.5).

Unlike in other areas of human activity, in mathematics the truth of claims has *the* central place; the Gricean maxim of quality is a *sine qua non*.[59] There are interesting aspects to how the other Gricean maxims regulate mathematical proofs. The maxim of quantity is manifested in the differences in level of detail, *granularity*, between various mathematical expositions. What is too much and too little information depends on the author's assumptions as to what the addressee knows – the common ground – and the purpose of the exposition. A mathematical textbook for novices differs in the level of detail from a scientific paper intended for experts; see also Chapter 2 (p. 60). Violation of the maxim may result in incomprehensible textbooks (overestimated assumed knowledge: too much information omitted) or in tedious mathematical articles (underestimated assumed knowledge: too much information included). In a tutoring setting, it is the tutor who, based on his assumptions on the student's knowledge, monitors the level of detail. A poorly performing student may be required to make some reasoning steps and justifications explicit which a good student may be allowed to skip; examples of tutor's reactions to the granularity of students' proof steps were shown in Section 3.2.1.5.[60]

There are two interesting aspects to *relevance* in the context of mathematics: one concerns the mathematical content and the other the informal language. Earlier in this section we said that the purpose of mathematical discourse is to communicate facts. In receiving mathematical discourse, the relevance of the presented content should be taken for granted: if something is said, it must be relevant and said for a reason. A mathematical proof does not admit of arbitrary facts if it is to fulfil its purpose of persuading, but rather only of those facts that make the addressee more convinced. An irrelevant assumption may lead to undesired implicatures. Halmos (1970, p. 138) illustrates this with the following example: '''If R is a commutative semisimple ring with unit and x and y are in R, then $x^2 - y^2 = (x - y)(x + y)$' The alert reader will ask himself what semisimplicity and a unit have to do with what he had always

[59] Paradoxically, the Quality Maxim is routinely flouted in one of the standard proof methods: proof by contradiction, in which a false statement is stated to be assumed to be true. This, however, serves the method's purpose of showing that the assumption is invalid by reaching a contradiction, thereby proving the original proposition.

[60] Granularity in human reasoning has been discussed by Hobbs (1985) and in proofs by Rips (1994). A computational framework for evaluating granularity for proof tutoring has been proposed in (Benzmüller and Vo, 2005; Autexier and Fiedler, 2006; Schiller et al., 2008).

thought was obvious.'' Likewise, irrelevant notation should be omitted and certain propositions, while true, may be unnecessary from the point of view of the argument. Students, however, do contribute irrelevant steps; we showed examples of such proof contributions in Section 3.2.1.5.[61]

The other aspect of relevance concerns the language of mathematical discourse. The formal language of mathematics, due to the nature of mathematics itself, is void of emphatic expressiveness and redundancy typical of natural language. Attitude or sentiment towards presented facts, any information which cannot be expressed in the formal language, or repetition of previously stated information is superfluous from the mathematical point of view.[62] However, informal mathematical discourse, especially in pedagogical context, does contain this kind of ``irrelevant'' content: statements may be reworded, paraphrased, or repeated for emphasis in order to facilitate understanding and recall or because of the limits of the addressee's attention span. Both the student and the tutor may explicitly linguistically mark *informationally redundant* contributions in order to bring out the fact that they are (or should be) already part of common ground.[63] Certain linguistic expressions may be used as part of mathematical ``jargon'' or for stylistic reasons to make the text ``read more naturally''. Linguistic means to convey this extra-mathematical content include adverbs, as in the previously quoted ``$A$ also $\subseteq B$'' (see p. 113) or discourse markers which do not contribute information on the logical structure of the proof, such as ``moreover'' or ``now''. From the point of view of mathematics, even naming theorems is unnecessary, but it makes communication of mathematics easier. There is no need for this kind of information in a formal representation in an automated reasoner; for computational processing of learner language this means that shallow methods could be used to identify such lexical material and to simplify the input preserving only the relevant content.

The maxim of manner is manifested in how proofs are presented. A remarkable property of formal mathematics is its precision. A formal proof contains no ambiguity, however, the symbolic notation may render it unreadable, a violation of the maxim of manner; recall the formal notation for sets of odd numbers and primes on page 114.[64] While an informal proof presented in natural language may contain ambiguities and irrelevant linguistic content (the kind mentioned above), it is typically cognitively easier to follow than a formal

[61] Computational aspects of evaluating relevance of proof steps are further discussed in (Benzmüller and Vo, 2005).

[62] (Except, of course, in formal systems in which formulas are explicitly reiterated.)

[63] See (Karagjosova, 2003) for a linguistic analysis and (Buckley and Wolska, 2007) for a computational model.

[64] As Halmos famously remarked ``The best notation is no notation'' (1970, p. 144). Gillman coined the term *symbolitis* for overuse of symbols in mathematical writing (Gillman, 1987, p. 7).

proof consisting of mathematical notation alone. The mode of presentation of mathematical discourse depends, in turn, on the purpose of the exposition and the intended addressee: In the tutoring setting different factors play a role than in textbooks or scientific publications. (Which brings us back to the motivation for collecting data specific to the tutoring setting; see Section 2.1 of Chapter 2.)

3.4 Conclusions

In the beginning of this chapter we presented mathematical language from the point of view of its properties as a sublanguage and as a kind of ''foreign'' language which students have to master in the course of learning mathematics. We have shown that phenomena typical to sublanguages, such as symbolic representations (Sections 3.2.1 and 3.2.2.1), deviant rules of grammar and recurrence of certain characteristic constructions (Section 3.2.2.3), and phenomena typical of various stages of mathematical cognitive development, such as imprecision of linguistic expression leading to ambiguity (Sections 3.2.2.4 and 3.2.2.5) or self-talk describing actions on the objects of discourse (Section 3.2.2.4), indeed occur in our corpora. Thus, modelling these phenomena in a language processing architecture for students' proofs should receive priority.

As we mentioned earlier, the examples in Section 3.2.2.1 show that a method of parsing symbolic expressions tightly interleaved with natural language is the fundamental functionality required for a computational interpretation module for mathematical language. Neither Zinn nor Natho offer a transparent computational solution to this problem although both do mention examples of such constructions. Zinn models constants and variables, effectively, as individual referents in DRSs with operators in complex terms and formulas as predicates in the DRSs' conditions and shows how to model only simple cases of appositive noun phrases and copula constructions in mixed language where the symbolic expression forms an atomic constituent (see Section 5.2 of (Zinn, 2004)). The approach lacks generalisation (individual atomic terms in the lexicon), modularity (single module for processing symbolic expressions and natural language), and is somewhat cumbersome by comparison with our approach proposed in (Wolska and Kruijff-Korbayová, 2004a). Natho claims to analyse the natural language and the symbolic language separately in mArachna (see (Natho, 2005, Section 3.3.3, discussion of Example 3.3.16, p. 121)). While examples of constructions with scope-bearing words interacting with parts of mathematical expressions are mentioned (for instance, ''Es gibt ein $e \in G$...'' (*There is an* $e \in G$); p. 143) no illustration of how they are handled is given and the result of analysis of the symbolic expressions is not integrated into the final interpretation result. In the ''Outlook'' section of (Natho et al., 2008), which appears to be the most recent publication of the mArachna group, the authors say that ''[including the content of formulas in the analysis and representation...] is not implemented. However, we are investigating an approach to rectify this deficiency. Therefore the use of a syntactical analysis, similar to those used in computer algebra systems in combination with contextual grammars (e.g. Montague grammars) to correlate the information given in a formula with information already provided in the surrounding natural language text, is proposed.'' However, no further details on how the Montague grammars would be realised are provided.

The presence of abbreviations, especially those with full stops, introduces extra complexity into computational sentence-boundary detection and word-tokenisation for mathematical discourse (Grefenstette and Tapanainen, 1994). A common approach is to create a lexicon of frequent abbreviations to help disambiguate occurrences of full stops (Reynar and Ratnaparkhi, 1997; Walker et al., 2001; Mikheev, 2002); see, for instance, (Schmid, 2000; Kiss and Strunk, 2006) for unsupervised approaches. Clearly, for mathematical discourse, a domain-specific abbreviations lexicon is needed.

The existence of two subsets of lexica in mathematical discourse, general and domain-specific (Section 3.2.2.2), motivate the need for modularity in the lexicon representation. First, a general lexicon should comprise general natural language vocabulary and the basic vocabulary of logic, necessary for any branch of mathematics. Second, separate domain-specific lexica should be accessed in specific contexts, depending on the mathematical domain of discourse. Both lexica should include a representation of multi-word expressions. A plausible approach would be to identify fixed phrases, such as ''dann und nur dann'' (*if and only if*), already at the preprocessing stage using shallow methods and to encapsulate them for further processing. Domain-specific lexica should, in turn, link to appropriate knowledge bases with formalised knowledge on the given domain.[65] The approach we propose in Chapter 5 is based precisely on this type of abstraction over domain-specific terminology; in Chapters 4 and 7 we show that even upon this lexical abstraction the students' language nevertheless proves surprisingly linguistically diverse.

Since imprecision phenomena are systematic and imprecision is cooperatively resolved, a computational interpretation component needs a representation of the imprecise concept names and an appropriate mapping to the possible specific mathematical interpretations. Notice moreover that this kind of ambiguity appears also in textbook discourse (recall, for instance, the previously quoted definition of set membership from (Bartle and Sherbert, 1982);

[65] MBase (Kohlhase and Franke, 2001) is an example of such a resource. See (Fiedler et al., 2002; Horacek et al., 2004) for a discussion on the interface issues.

see p. 96 of this chapter) which all the more motivates this as a basic requirement for a computational processing architecture. In order to account for discourse references to parts of mathematical expressions, three issues have to be taken into account: First, the set of substructures of mathematical expressions which are relevant to resolving references must be identified, for instance, by a systematic corpus study and by observations on common usage of references to specific mathematical expression parts. Second, symbolic representations of these entities must be included in the domain knowledge representation. And third, the substructure entities must be available for reference in the discourse model. An anaphor resolution algorithm needs to identify plausible reference scopes within complex symbolic expressions within which antecedent search should be performed. We address some of these issues in Section 6.3. Aside from cooperative interpretation of imprecise language, cooperative interpretation of ill-formed expressions is needed. The fact that the tutors hardly ever explicitly requested that errors in the symbolic language be corrected suggests that focus should be on problem solving; that is, an intelligent tutoring system should be capable of cooperative reaction even if formulas are ill-formed. In Section 6.4 we show results of a pilot study on error correction based on the common sources of errors showed in Section 3.2.1.5.

Finally, frequent occurrence of complex clause structures in paratactic and hypotactic configurations calls for a grammar formalism in which complex multi-clause utterances could be modelled with sufficient generality. (In a context-free grammar, every instance of clause ordering would have to be modelled explicitly in order to obtain all the possible parses; a suboptimal solution.) For German specifically, the different word orders in main clauses and subordinate clauses need to be modelled in a systematic way. This requires an expressive enough grammar formalism with a syntax–semantics interface capable of constructing appropriate semantic representations. Moreover, structurally ambiguous readings (Section 3.2.2.3) need to be represented (be it in a compact underspecified way or by enumerating alternative parses) since the linguistic processing module is not in a position to disambiguate the intended reading. In Chapter 5 we motivate the choice of Combinatory Categorial Grammar as a grammar formalism which enables perspicuous modelling of various phenomena observed in the corpora, in Chapter 6 we show how we model basic German syntax relevant for mathematical discourse, and finally, in Chapter 7 we show that categorial grammars we have developed based on our data provide better linguistic generalisations than context-free grammars, while remaining at manageable levels in terms of ambiguity. Before presenting our approach to modelling language phenomena, in the next chapter, we analyse the diversity of students' productions in quantitative terms.

Chapter 4

Quantitative analysis of the students' language

In this chapter we quantitatively analyse the diversity in the students' language. Both corpora described in Chapter 2 are used as data. The analysis is performed at a ''shallow'' level in the sense that we only look at linguistic verbalisation patterns and at the patterns' shallow (quantitative) characteristics. The purpose of the analysis is to verify two hypotheses: The first hypothesis stems from prior claims made based on textbook mathematical discourse which suggested that the language of proofs tends to be simple and repetitive (Zinn, 2004; Natho, 2005); we postulate, to the contrary, that the students' language is complex and diverse. The second hypothesis is that the language of students' interaction is influenced by the style of presentation of the study material (see ''Study material'' in Section 2.4.3). The analysis is moreover intended to inform and motivate the choice of computational input processing methodology for a tutoring system for mathematical proofs.

We start by classifying the students' utterances within their dialogue context. Next, we outline the preprocessing procedures. The results are presented as follows: First, the students' language is characterised in terms of linguistic ''modality'' (natural language vs. symbolic notation). The binary relations corpus is characterised in terms of differences in the language between the two study material conditions. Then, we look at the distribution of utterance types in both corpora. Proof contributing utterances are further analysed with respect to their function in the proof under construction (proof steps, declarations of proof strategy, etc.) and the type of content verbalised in natural language (logical connectives only, domain-specific vocabulary, etc.) Linguistic diversity along these dimensions is quantified in terms of type–token ratios over the normalised linguistic patterns, frequency spectra, and pattern-vocabulary growth curves. Material presented in this chapter appeared in (Wolska and Kruijff-Korbayová, 2006a; Wolska, 2012).

C-I

S1: Wenn $A \subseteq K(B)$, dann $A \cap B = \emptyset$
(*If $A \subseteq K(B)$, then $A \cap B = \emptyset$*)

...

S5: in $K(B)$ sind alle x, die nicht in B sind
(*in $K(B)$ are all x which are not in B*)

...

C-II

S1: Ich moechte zunaechst $(R \circ S)^{-1} \subseteq S^{-1} \circ R^{-1}$ beweisen
(*First I would like to prove $(R \circ S)^{-1} \subseteq S^{-1} \circ R^{-1}$*)

S2: Sei $(a, b) \in (R \circ S)^{-1}$
(*Let $(a, b) \in (R \circ S)^{-1}$*)

...

S6: Nach der Definition von $\circ$ folgt dann (a, b) ist in $S^{-1} \circ R^{-1}$
(*By definition of $\circ$ it follows then that (a, b) is in $S^{-1} \circ R^{-1}$*)

...

S8: Der Beweis geht genauso wie oben, da in Schritt 2 bis 6 nur Aequivalenz umformungen stattfinden
(*The proof goes exactly as above since in step 2 to 6 there are only equivalences*)

S9: wie kann ich jetzt weitermachen?
(*how can I continue now?*)

...

S11: 1. Fall: Sei $(a, b) \in R$
(*1. Case: Let $(a, b) \in R$*)

S12: Ich habe mich vertippt. Korrektur: Sei $(a, z) \in R$
(*I made a typo. Correction: Let $(a, z) \in R$*)

...

S17: Ich habe gezeigt: $(a, b) \in (R \cup S) \circ T \Rightarrow (a, b) \in R \circ T \vee (a, b) \in S \circ T$
(*I have shown: $(a, b) \in (R \cup S) \circ T \Rightarrow (a, b) \in R \circ T \vee (a, b) \in S \circ T$*)

...

S24: Dann existiert ein z, so dass $(a, z) \in (R \cup S)$ und $(z, b) \in T$
(*Then there exists an z such that $(a, z) \in (R \cup S)$ and $(z, b) \in T$*)

S25: Nach Aufgabe A gilt $(R \cup S) \circ T = (R \circ T) \cup (S \circ T)$
(*By Exercise A $(R \cup S) \circ T = (R \circ T) \cup (S \circ T)$ holds*)

...

S29: Da die Mengenvereinigung kommutativ ist, koennen wir dieses in student 25 einsetzen und erhalten die Behauptung
(*Since set union is commutative, we can use what's in student 25 and obtain the theorem*)

...

Figure 4.1: Examples of students' utterances from both corpora

- Solution-contributing
  - Proof contribution
    - Proof step
    - Proof strategy
    - Proof structure
    - Proof status
  - Meta-level
    - Self-evaluation
    - Restart
    - Give up
- Other
  - Help request
  - Yes/No
  - OK
  - Agree
  - Address
  - Answer
  - Cognitive state
  - Self-talk
  - Session
  - Discourse marker
  - Politeness/Emotion/Attitude (P/E/A)
- Uninterpretable

Figure 4.2: Typology of students' utterances

4.1 Utterance typology

Students' contributions in tutoring interactions may fulfil several functions. Examples of dialogues from both corpora were already shown in Chapter 2 (pp. 82, 83), however we did not point out different functional types of students' utterances. Figure 4.1 shows two further excerpts which exemplify utterance types found in our data. As the examples illustrate, students contribute not only proof steps – complete or incomplete, as in C-I S5 (a justification of the statement is not given), explicit or implicit, as in C-II S8 (instead of a proof step, a high-level description of a set of steps is given) – but also other content which adds to the solution indirectly, as in C-II S1 (a solution strategy is described) or C-II S11 (a proof structure to follow, case distinction, is signalled) or which does not add to the solution at all, as in C-II S9 (help is requested).

In order to investigate linguistic diversity of students' language at a level corresponding to different contribution types, we designed a typology of students' utterances based on the two corpora. The present classification builds on previously proposed dialogue move taxonomies for tutorial dialogue (Marineau et al., 2000; Campbell et al., 2009; Becker et al., 2011) and has been adapted specifically for the proof tutoring domain based on the analysis of our data. The classifications by Marineau et al., Campbell et al., and Becker et al. model students' contributions at a high-level and are too coarse-grained at the task-level (here: proving) for our purposes. Our previous classification presented in (Wolska and Buckley, 2008) was designed with dialogue modelling in mind, rather than analysis of language diversity or input interpretation, and it does not make distinctions which are relevant here either.

The classification we propose, shown in Figure 4.2, has a shallow hierarchical structure focusing on *Solution-contributing* content. All the non-solution contributing utterances are grouped into one category, *Other*, with an extra class *Uninterpretable* for utterances whose semantics or pragmatic intent could not be interpreted; for instance, because they were cut off mid-utterance. The distinction between the *Solution-contributing* class and *Other* is that with *solutions* the student is adding information to the solution he is constructing, be it by contributing a step or steps, changing the meta-level status of the solution (for instance, stating that a new attempt at a solution will be made) or by signalling a revision or an evaluation of an already contributed solution. The *Other* class may also comprise utterances which express students’ knowledge, but only those explicitly elicited by the tutor (*Answer*). The classification of utterances which do not contribute solution steps is coarse-grained for two reasons: First, we are mainly interested in the analysis of students’ proof language. Second, as will become clear in Section 4.3.3 the frequency of the *Other* utterance types is in general low; with the exception of *Help requests*.

The *Solution-contributing* utterances are subdivided into two classes: *Proof contributions* with four subclasses (*Proof step*, *Proof strategy*, *Proof structure*, *Proof status*) and *Meta-level* contributions with three classes (*Self-evaluation*, *Restart*, *Give up*). The classes are described below and exemplified:

Proof step Contributes a proof step or part of a proof step. Examples of utterances of this type include C-I S1 and S5 and C-II S2 and S6 in Figure 4.1, as well as, for instance, the utterance ‘‘Begruendung: $A \subseteq (U \setminus B)$’’ (*Justification: . . .*) which specifies only the justification of a proof step.

Proof strategy States a solution strategy already adopted or about to be adopted. Examples include ‘‘Ich benutze das Extensionalitaetsprinzip’’ (*I’m using the Extensionality Axiom*), ‘‘Beweis durch $\subseteq$ und $\supseteq$’’ (*Proof by $\subseteq$ and $\supseteq$*).

Proof structure Signals the structure of the solution being constructed, as in C-II S1 in Figure 4.1 or ‘‘Ich mache eine Fallunterscheidung’’ (*I’m making a case distinction*), ‘‘Hinrichtung’’ (*Forward direction*).

Proof status Signals the status of a (partial) solution: ‘‘Damit ist eine Inklusion bewiesen’’ (*And so one subset relation is shown*), ‘‘q.e.d.’’

Self-evaluation States an evaluation of own step: ‘‘Ich habe mich vertippt’’ (*I’ve made a typo*), ‘‘Schwachsinn’’ (*Nonsense*), or ‘‘Korrektur’’ (*Correction:*).

Restart Signals a new attempt at a proof: ‘‘neuer Anfang’’ (*new start*) or ‘‘Wieder von vorne’’ (*Once again from the beginning*).

Give up Signals abandoning the solving task: ‘‘Ich moechte die Antwort wissen’’ (*I would like to know the solution*), ‘‘ich gebe auf’’ (*I’m giving up*).

The non-solution-contributing utterances are subdivided into 11 subclasses:

Help request Requests assistance explicitly: ‘‘Ich brauche einen Tip’’ (*I need a hint*), ‘‘bin ich auf dem richtigen Weg?’’ (*am I on the right track?*)

Yes/No A ‘‘yes’’ or ‘‘no’’ answer

OK A simple acknowledgement: ‘‘okay’’

Agree Expresses agreement: ‘‘du hast natuerlich recht’’ (*of course you’re right*)

Address Provides a *non-elicited* reaction to a contribution: ‘‘Das beantwortet meine Frage nur zur Haelfte!’’ (*This answers my question only halfway!*), ‘‘Die Klammer koennte ich nach meinem Dafuerhalten auch ganz woanders setzen!’’ (*The bracket could just as well be in a different place if you ask me!*)

Answer Provides an *elicited* non-Yes/No answer to a question:
T: ‘‘Was sind moegliche Eigenschaften von binaeren Relationen?’’ (*What are possible properties of binary relations?*)
S: ‘‘symmetrisch’’ (*symmetric*)

T: ‘‘Was bedeutet die Variable x bei Ihnen?’’ (*What does the variable x mean?*)
S: ‘‘x hat zwei Bedeutungen es kommt in zwei verschiedenen Mengen vor’’ (*x has two meanings it is in two different sets*)

Cognitive state Expresses the state of knowledge or understanding: ‘‘ich weiss nicht, was ich mit den Tips anfangen soll’’ (*i don’t know what i can do with these hints!*), ‘‘Das weiss ich’’ (*I know that.*)

Self-talk Expresses an unelicited comment: ‘‘Fraglich was ist unterschied zwischen $=$ und $\cap$’’ (*The difference between $=$ and $\cap$ is questionable*), ‘‘Muss mit der Differenz zusammenhaengen’’ (*Must have something to do with the difference.*)

Discourse marker The utterance has a sole discourse marker function: ‘‘Na ja’’ (*Right...*), ‘‘Also gut’’ (*Good then.*)

Session Expresses a meta-level statement related to the tutoring session itself: ‘‘Allerdings ist Aufgabe E (wie Du es bezeichnest) bei mir Aufgabe A!’’ (*Actually Exercise E (as you call it) is called Exercise A here!*), ‘‘wie waere es, Aufgabe W nach hinten zu verschieben und mit Aufgabe A zu starten?’’ (*how about postponing Exercise W and starting with A?*)

Politeness/ Emotion/ Attitude (P/E/A) Expresses politeness in a conventional way or the speaker’s emotion or attitude: ‘‘Sorry!’’, ‘‘Ich werde Dich im Geschaeft umtauschen’’ (*I will exchange you at the shop!*), ‘‘Keine PAnik’’ (*Don’t panic*), ‘‘NERV!’’ (*[annoyance]*)

Note that the classification can be mapped to previously proposed classifications of dialogue actions in tutoring. For instance, the category *Proof contribution* corresponds to *Assertions* in (Marineau et al., 2000), *Contribute domain content* in (Wolska and Buckley, 2008), *Information Exchange : Assert* in (Becker et al., 2011), and comprises the categories *Solution-step* and *Solution-strategy* from (Buckley and Wolska, 2008b). Following the general scheme proposed in (Campbell et al., 2009) our class of *Proof contributions* which do not explicitly signal informational redundancy would be coded in the *Novelty* dimension for steps contributing new content (C-II S17 is a counter-example) and in the *Motivation* dimension as *Internal* or *External*, depending on whether they have been elicited by the tutor. Utterances in the *Motivation: External* category would be found, among others, in our *Answer* category.

The presented utterance typology has been developed by an exhaustive analysis of all students' utterances in all dialogues from the two corpora and based on the insights from applying our previous tutorial dialogue coding scheme presented in (Buckley and Wolska, 2008b) and its generalisation presented in (Wolska and Buckley, 2008).[1] Over multiple annotation cycles, we arrived at a reference annotation which will be used in the following sections. At present, the utterance typology has not been applied by independent annotators and evaluated in terms of inter-coder agreement. Notice, however, that classification of utterances into the critical categories, the solution-contributing classes, does not require linguistic knowledge, but rather knowledge of mathematics, in particular, proof methods. Assuming clear understanding of proof-related notions, no ambiguity is expected. Therefore, multiple annotations have not been performed. Moreover, the classification has been designed in such way that cross-category confusion is minimised. Among the *Other* class, *Help request*, *Agree*, *Cognitive state*, *Session*, *Yes/No*, *OK*, *Discourse marker* are clear-cut. The first four are semantically clearly distinguishable, while the latter three can be considered for the most part lexically defined. Within the remaining four classes confusion may arise between *Address* and *Self-talk*, however, there were only two instances of the latter and the distinction was made only because in the dialogue context the *Self-talk* utterances appear to refer to the students' own contribution and have a character of think-aloud comments, whereas *Addresses* tend to refer to the tutors' contributions. The distinction between the elicited *Answer* and the non-elicited *Address* appears clear-cut. Utterances such as ''The hint was rather lousy'' could be mistakenly classified as *P/E/A* (that is, interpreted as expressing an attitude towards the tutor's hint, a plausible alternative), however, this can be avoided by placing the decision question targeting the *Cognitive state* class higher in the annotation

[1] Utterance identification guidelines we followed will be presented in the next section.

scheme's decision tree. Within the *Solution contributing* utterances, *Meta-level* types are clear-cut. A confusion may arise between *Proof strategy* and *Proof structure* if an annotator should not understand the notion of proof strategies, however, again, the frequency of the classes is low relative to the frequency of the majority classes, *Proof step* and *Help request*.

4.2 Preprocessing

Three preprocessing transformations have been performed on the students' data before the analysis: First, utterance boundaries have been identified, second, mathematical expressions have been normalised, and third, a number of textual normalisations have been performed with the goal of abstracting over domain-specific terminology and eliminating spelling and writing mechanics differences. Details of corpus preprocessing are outlined below.

4.2.1 Turn and utterance preprocessing

Turns in both corpora have been sentence-tokenised based on a standard set of end-of-sentence punctuation. Word-tokenisation was performed using a standard tokeniser. The outputs were verified and manually corrected where necessary.

Turns were then segmented into utterances. While a sentence is typically defined as a unit of speech containing a subject and a predicate, there is no precise linguistic definition as to what constitutes an utterance. Broadly understood, an utterance is an intentional, meaningful communicative act in an interaction. An utterance may consist of a word, a phrase, or a complex sentence with embedded clauses. It may form a complete turn, but a turn may also consist of more than one utterance. For the purpose of this study, in particular also for the purpose of utterance type annotation, the notion of an utterance was operationalised as follows:

- An utterance never spans more than one turn or one sentence;
- Multiple clauses conjoined with conjunctions (''und'' (*and*), ''oder'' (*or*), ''aber'' (*but*), ''weil'' (*because*), ''für (*for*), ''also'' (*so*), ''wenn'' (*if*), ''als''/''wann'' (*when*), etc.) were considered one utterance;
- Multiple clauses conjoined without conjunctions were considered separate utterances;
- ''If-then'' constructions, also omitting the words ''if'' or ''then'', were considered a single utterance;

- The following non-sentential fragments, not containing a subject, were considered utterances: noun phrases, discourse markers (also inserts, such as ‘‘acha’’, ‘‘oh’’, ‘‘naja’’, ‘‘schoen’’ (*nice*)), colloquial subject-drop phrasings in indicative and interrogative mood, ellipted questions (for instance, ‘‘Fertig?’’ (*Done?*)), politeness phrases (such as ‘‘sorry’’, ‘‘Danke’’), exclamatives (‘‘Weitere Hilfe!’’ (*Further help!*)), non-sentential answers to questions, including acknowledgements, for instance, ‘‘ok’’, ‘‘klar’’ (*that's clear*), as well as yes/no answers.

Examples of tokenised multi-utterance turns from Figure 4.1 are shown below (vertical bars mark token boundaries, $\langle u\rangle$ and $\langle /u\rangle$ utterance boundaries; here and further: ‘‘O’’ labels original utterances, ‘‘P’’ preprocessing results):

O: Dann gilt auch : Alle x, die in B sind, sind nicht in A
P: $\langle u\rangle$|Dann|gilt|auch|:|Alle|x|,|die|in|B|sind|,|sind|nicht|in|A|$\langle /u\rangle$
O: 1. Fall: Sei $(a,b) \in R$
P: $\langle u\rangle$|1.|Fall|:$\langle /u\rangle$ $\langle u\rangle$Sei|$(a,b) \in R$|$\langle /u\rangle$

O: Ich habe mich vertippt. Korrektur: Sei $(a,z) \in R$
P: $\langle u\rangle$|Ich|habe|mich|vertippt|.|$\langle /u\rangle$ $\langle u\rangle$|Korrektur|:|$\langle /u\rangle$
$\langle u\rangle$|Sei|$(a,z) \in R$|$\langle /u\rangle$

4.2.2 Preprocessing mathematical expressions

In both corpora, mathematical expressions were identified semi-automatically, using a regular expression grammar. The grammar comprised a vocabulary of letters, mathematical symbols (unicode or LaTeX), brackets, braces, delimiters, etc. The parser's output was manually verified and corrected where necessary.[2] The quantitative analyses were conducted based on turns and utterances in which the identified mathematical expressions have been substituted with a symbolic token MATHEXPR. As we will show in Chapter 5 utterances preprocessed this way can be parsed using a lexicalised grammar if the information on the expression's type – term or formula – is known. With this in mind, we therefore also classify the symbolic expressions into one of the following categories: (i) atomic terms: VAR, for set, relation, or individual variables, (ii) non-atomic terms: TERM (object-denoting expressions) or _TERM_ (term-forming operation symbols appearing in isolation,

[2] We do not report precision results on mathematical expression identification and parsing as it is not the focus of this work. It is assumed that an end-to-end system provides an entry method for mathematical expressions which would enable clear, possibly real-time, identification of mathematical expressions. This could be accomplished by explicitly defining ‘‘math mode’’ delimiters, for instance, as key combinations indicating the start and end of mathematical expression strings or as textual delimiters analogous to the $-symbols in LaTeX.

as in the example utterance (8) in Section 3.2.2.3 of the previous chapter; underscores denote non-realised (missing) arguments), etc. and (iii) formulas, FORMULA, for truth-valued statements, _FORMULA_ (statement-forming operators appearing in isolation), etc. Examples of utterances from Figure 4.1 before and after mathematical expression preprocessing are shown below:

O: Da $A \subseteq K(B)$ gilt, alle x, die in A sind sind auch nicht in B
P: Da MATHEXPR$_{\text{FORMULA}}$ gilt, alle MATHEXPR$_{\text{VAR}}$, die in
MATHEXPR$_{\text{VAR}}$ sind sind auch nicht in MATHEXPR$_{\text{VAR}}$

O: Nach der Definition von $\circ$ folgt dann (a, b) ist in $S^{-1} \circ R^{-1}$
P: Nach der Definition von MATHEXPR $_{\text{\_TERM\_}}$ folgt dann
MATHEXPR $_{\text{TERM}}$ ist in MATHEXPR $_{\text{TERM}}$

4.2.3 Textual normalisations

Following extensive research into the properties of spoken and written discourse (Chafe and Tannen, 1987; Biber, 1988), recent studies on computer-mediated communication (CMC) – or electronic discourse more generally – have shown that, much like spoken language differs from written language, the language of type-written computer-mediated communication shares some properties with spoken language, however, it also possesses textual and linguistic characteristics which are not typical of standard written language (Maynor, 1994; Crystal, 2001; Hård af Segerstad, 2002; Baron, 2003). Among those non-standard characteristics are: frequent use of abbreviations and acronyms, use of all capitals or all lower-case script, extensive use of certain punctuation marks or lack or incorrect (random) use of punctuation (for instance, excessive use of the exclamation mark, lack of or incorrect use of commas, lack of valid end-of-sentence punctuation), and the use of emoticons. Type-written tutorial dialogue shows qualities which are found both in spoken and written language and those of CMC. It is prone to textual ill-formedness due to the informal setting and the telegraphic nature of the linguistic production.

In order to avoid the effects of CMC-specific qualities of the learners' productions at the utterance-level, prior to the quantitative analysis learners' utterances were normalised with respect to certain writing mechanics phenomena (alternative spelling variants, capitalisation, punctuation) and with respect to the wording of common abbreviations. A number of lexical normalisations were performed on lexemes and phrases in order to avoid spurious diversity due to domain-specific terminology and task-specific contextual references. Different lexical realisations of single and multi-word domain terms and conventional speech acts were substituted with symbolic tokens representing their lexical, in case of the former, or communicative, in case of the latter, types. Discourse-specific references were likewise normalised. General language expressions and references other than those mentioned below as well as general mathematical terms (such as ‘‘assumption’’, ‘‘definition’’, for instance) were not normalised. All the normalisations were performed semi-automatically; the results of a preprocessor were reviewed and corrected manually in case of errors. Details of textual normalisations are summarised below.

Spelling The German Umlaute were replaced with their underlying vowels and an ‘‘-e’’ and *eszett* ligatures with double ‘‘s’’. Misspellings were identified and corrected using German aspell, a Linux spell-checker, whose dictionary has been extended with a custom dictionary of relevant domain terms.

Punctuation Repeated consecutive occurrences of the same punctuation symbols were replaced with a single occurrence (‘‘!!!’’ with ‘‘!’’; ‘‘....’’ with ‘‘.’’, etc.) Punctuation in abbreviations, missing or incorrect, has been normalised (‘‘bzw.’’ for ‘‘b..zw’’ ‘‘d.h.’’ for ‘‘d.h’’, etc.). In the final analyses intra-sentential and end of sentence/utterance punctuation was ignored.

Abbreviations Upon correcting punctuation, different correct and incorrect lexical variants of common abbreviations were substituted with symbolic tokens. These included, BSP for different spelling and capitalisation variants of ‘‘z.B.’’ (*e.g.*), BZW for ‘‘bzw.’’ (*respectively*), OBDA for ‘‘o.B.d.A.’’ (*without loss of generality*), DH for ‘‘d.h.’’ (*that is*), QED for ‘‘q.e.d.’’, ST for ‘‘s.t.’’ (*such that*), OK for ‘‘ok’’, ‘‘oki’’, ‘‘Okay’’, etc.

Common speech acts and inserts Conventional expressions of gratitude, such as ‘‘Danke’’, ‘‘VIELEN DANK’’ and apologies, for instance, ‘‘Tut mir leid’’, ‘‘Sorry’’, ‘‘Verzeihung’’, were substituted with tokens THANKYOU and APOLOGY, respectively. ‘‘Ja’’/‘‘Nein’’ responses were substituted with YESNO. Conversational inserts and other discourse markers, such as ‘‘So’’, ‘‘Na ja’’, were substituted with DISCOURSEMARKER.

Domain terms and domain-specific references Different lexical variants of nominal and adjectival domain terms which were included in the preparatory material have been mapped to a single form, DOMAINTERM. If single-word domain terms were part of a multi-word term which can be considered a named entity, the multi-word term was normalised. For instance, ‘‘DE-MORGAN-1’’, ‘‘DeMorgan-1’’, ‘‘DeMorgan-Regel-1’’, ‘‘de morgan regel 2’’ all mapped to DOMAINTERM, as did ‘‘Distributivitaet von Vereinigung ueber den Durchschnitt’’ as a multi-word term (a name of a statement/theorem), as well as ‘‘symmetrisch’’ as a single-word term. Non-deictic references to proof exercises, such as ''Aufgabe W'' (*Exercise W*), theorems provided in the preparatory material, such as ''Theorem 9'' or ''9'', parts of proof structure, such as ''Schritt 1'' (*Step 1*), or turns in the dialogue history, such as ''Student 25''[3], were mapped to the token REFERENCE. Deictic references, such as ''obiges'' (*the above*) were not normalised.

Different conventional wordings used to signal the end of a proof, such as ''quod erat demonstrandum'', ''was zu zeigen war'' (*which was to be shown*), ''woraus der beweis folgt'' (*from which the proof follows*), ''Damit ist der Beweis fertig'' (*which completes the proof*), etc., were mapped to the token corresponding to the ''q.e.d.'' abbreviation, QED.

Capitalisation The analyses were performed on corpus utterances normalised as above with *case-insensitive* matching. Examples of utterances from Figure 4.1 preprocessed as outlined in this section are shown below:

dann existiert ein MATHEXPR so dass MATHEXPR und MATHEXPR
nach REFERENCE gilt MATHEXPR
da DOMAINTERM DOMAINTERM ist koennen wir dieses in
REFERENCE einsetzen und erhalten die Behauptung
nach REFERENCE und REFERENCE gilt MATHEXPR

Further in this chapter we will refer to students' contributions preprocessed in this way as ''verbalisation patterns'', ''utterance patterns'', or simply ''patterns''. Whenever we say ''turns'' or ''utterances'' we mean turns or utterances preprocessed as described here.

4.3 Diversity of verbalisation patterns

We begin the quantitative analysis with a high-level overview of the amount of natural language in the students' contributions by looking at the distribution of turns and utterances formulated using mathematical symbols alone, natural language alone, and using natural language interleaved with mathematical symbols and at differences in the amount of natural language between the two study material conditions in C-II. Next, we focus on utterances formulated using *some* natural language. We first look at the distribution of utterance types in the two corpora. Then we take a closer look at *Proof contributions*, in particular the *Proof step* category, in terms of the type of verbalised content. We summarise the most frequently occurring linguistic forms – *verbalisation patterns* – by

[3] References of this form are artefacts of our dialogue display interface. In the dialogue history, student turns were numbered and labelled ''Student 1'', ''Student 2'', etc. while tutor turns were labelled ''Tutor 1'', etc.

category, and analyse the growth of diversity of forms with the increasing corpus size. In all analyses we consider the two corpora separately and a corpus consisting of the two corpora combined into one data set (C-I&C-II).

Two frequency counts are reported in the tables throughout this chapter: ‘‘Total’’ denotes the number of turn/utterance instances (that is, ‘‘vocabulary size’’; ‘‘vocabulary’’ here are verbalisation patterns). ‘‘Unique’’ is the number of *distinct* patterns. The ratio of these measures is known as ‘‘type–token ratio’’. The two raw frequencies rather than the summarised measure are provided because the number of tokens is different for each cell in the tables, so the raw counts are more informative. We also plot frequency spectra. Spectra visualisations are typically used with word frequencies to show a frequency distribution in terms of the number of types by frequency class; a frequency class is a set of (sets of) instances with the same number of occurrences in the data. In other words, they show how many *distinct types* (y-axis) occur once, twice, and so on (x-axis), thus revealing the degree of skewness of the types distribution; the earlier the tail with y around 1 starts, the more idiosyncratic the types. We use verbalisation patterns as units of analysis.[4]

4.3.1 Mathematical symbols vs. natural language

As the first approximation of linguistic variety in learner proof discourse, we analyse the students' contributions in terms of the two types of content modalities: natural language and symbolic expressions. Table 4.1 (p. 161) shows the distribution of turns and utterances in both corpora with respect to natural language and symbolic content. ME denotes turns and utterances consisting of symbolic expressions alone, NL those consisting of natural language alone (as in C-II S8 or C-II S29), and ME & NL consisting of natural language interleaved with mathematical expressions (C-I S1, C-II S6, C-II S24).

The majority of turns and utterances contain some natural language (turns: 54% NL/ME&NL vs. 46% ME in C-I and 70% vs. 30%, respectively, in C-II; utterances: 57% NL/ME&NL vs. 43% ME in C-I and 73% and 27%, respectively, in C-II). There are 640 *turn*-level NL/ME&NL patterns in C-I and C-II considered in isolation and 626 in C-I&C-II and 728 *utterance*-level patterns in C-I and C-II in isolation vs. 700 in C-I&C-II. This means that there are only 14 NL/ME & NL turn-level patterns and only 28 utterance-level patterns which occur both in C-I and C-II. Verbalisation patterns which occurred in both corpora are shown in Table 4.2 (p. 162). Overall, 69% of the utterances in C-I&C-II contain some linguistic material, among which there

[4] The zipfR package (Evert and Baroni, 2007) was used to generate frequency spectra. Only the first 15 frequency classes are shown since frequency of the larger classes oscillated between 0 and 5.

Table 4.1: Descriptive information on learner proof discourse in terms of content modality: symbolic (ME), natural language (NL), and mixed (ME&NL)

		C-I	C-II	C-I&C-II
		Unique / Total	Unique / Total	Unique / Total
Turns		147 / 332	497 / 927	628 / 1259
	ME	2 / 153	2 / 274	2 / 427
	NL	34 / 51	134 / 162	163 / 213
	ME & NL	111 / 128	361 / 491	463 / 619
Utterances[1]		200 / 443	531 / 1118	702 / 1561
	ME	2 / 189	1 / 300	2 / 489
	NL	64 / 92	185 / 278	240 / 370
	ME & NL	134 / 162	345 / 540	460 / 702

[1] A single occurrence of an utterance consisting of a question mark alone (in C-II) is included in the NL category.)

are 700 distinct patterns. There is proportionally more natural language in C-II even though, as we will show in the next section, the participants in the formal material condition were less verbose than those in the verbose condition.

4.3.2 The effect of the study material presentation

Recall that the second experiment was set up to test a hypothesis concerning the students' language production. The hypothesis was that the presentation of the study material, formal vs. verbose, would influence the students' language, resulting in proofs formulated using mainly symbolic language (formal) or using mainly mixed or natural language (verbose condition). C-II comprises 927 students' turns (Table 2.2), 471 in the FM group and 456 in the VM group.

Measures In order to investigate the differences in dialogue styles with respect to language production we first compared general dialogue characteristics in terms of content modality (mathematical expressions, ME, vs. mixed language, ME & NL, vs. natural language alone, NL) and session lengths measured as total number of turns. Then, we compared the following *session* and *turn* characteristics: number of mathematical expressions (ME tokens), number of natural language tokens (NL tokens), and mathematical expression lengths in characters (ME-length). By ME tokens we mean the number of mathematical expressions normalised as described earlier; counted were occurrences of formulas, terms, and single character tokens intended to represent relation or set symbols. ME-lengths were computed by counting all characters intended

Table 4.2: Verbalisation patterns found in both corpora

Solution-contributing patterns	Other
es gilt MATHEXPR	was ist MATHEXPR
dann ist MATHEXPR	ich brauche hilfe
also ist MATHEXPR	warum nicht
MATHEXPR und MATHEXPR	YESNO
daraus folgt dass MATHEXPR	OK
daraus folgt MATHEXPR	THANKYOU
damit ist MATHEXPR	APOLOGY
damit gilt MATHEXPR	DISCOURSEMARKER
somit ist MATHEXPR	
dann ist MATHEXPR und MATHEXPR	
das heisst MATHEXPR	
aus MATHEXPR folgt MATHEXPR	
MATHEXPR ist DOMAINTERM	
also gilt MATHEXPR und MATHEXPR	
also gilt auch MATHEXPR	
MATHEXPR ist DOMAINTERM von MATHEXPR	
also ist auch MATHEXPR	
das gleiche gilt fuer MATHEXPR	
DOMAINTERM	
QED	

to form a mathematical expression, including punctuation and single character tokens for variables and constants; ill-formed expressions were included.[5]

If parametric assumptions were met (as per Shapiro-Wilk and Levene tests), two-sided independent samples t-test was used to compare the means of the above-mentioned measures between groups; otherwise the Mann-Whitney-Wilcoxon test was used. The significance level was set at 0.05. Statistical differences are marked in bold; standard deviations in parentheses.

Turns by content modality Table 4.3 shows the absolute numbers and proportions of students' turns which consisted of mathematical expressions alone (ME), natural language alone (NL), and of the mixed language (ME & NL). The largest proportion of turns in the FM-group consisted of mathematical expressions alone, while in the VM-group of a mixture of mathematical

[5] The figures differ from those in (Wolska and Kruijff-Korbayová, 2006a) for two reasons: here we excluded turns generated automatically when the student clicked on the next exercise or ended the session and we include punctuation as tokens. The overall comparison results are not affected.

Table 4.3: Distribution of students' turns by content modality and study material

Content modality	FM-group (N=471)	VM-group (N=456)
ME	200 (42%)	74 (16%)
ME & NL	184 (39%)	307 (67%)
NL	87 (18%)	75 (16%)

Table 4.4: Means and standard deviations of session lengths

Measure	FM-group (N=471)	VM-group (N=456)
Session length	48.50 (15.89)	55.06 (22.78)

expressions and natural language. Also, the proportion of turns consisting of symbolic material alone was larger in the group presented with formalised material; 42% of all student turns in the FM-group vs. 16% in the VM-group.

Session length Table 4.4 shows descriptive information on session lengths. The dialogues in the VM group tended to be longer, however, the difference in the session lengths between the two conditions is not statistical (p>0.10).

Students' language production Finally, we compare the students' language production per session and per turn in detail. The average number of mathematical expression tokens per session was 35.11 (18.67) and the average number of natural language tokens was 119.73 (98.82). The average mathematical expression length in the dialogues was 17.35 (20.55) characters.

Table 4.5 summarises two sets of measurements: mean numbers of natural language tokens (NL tokens), mathematical expression tokens (ME tokens), and mean mathematical expression length (ME-length). The top part of the table shows the averages for the entire sessions (per session). The bottom part shows the same measurements averaged for turns (per turn).

While there was little difference between the VM- and FM-groups in the number of turns which contained natural language words alone (see Table 4.3), the average number of natural language words per session and turn is higher in the VM-group (p<0.05). The average number of mathematical expressions per session and turn was also higher in the VM-group (p<0.01), however, the average math expression length was significantly higher in the

Table 4.5: Means and standard deviations of the language production measures

	Measure	FM-group	VM-group
Per session	ME tokens	26.95 (10.49)	**44.70** (21.74)
	NL tokens	93.00 (89.03)	**151.18** (103.03)
	ME-length	**27.79** (17.64)	12.45 (8.85)
Per turn	ME tokens	1.14 (1.12)	**1.67** (1.70)
	NL tokens	3.95 (5.65)	**5.63** (6.02)
	ME-length	**32.53** (27.71)	15.69 (14.16)

FM-group ($p<0.01$). Note that the longest mathematical expression had 145 characters in the FM-group and 110.00 in the VM-group. The relatively large ME lengths may be an artefact of the interface's copy–paste mechanism. Students tended to copy formulas from the previous dialogue or the study material into their input-line and modified or extended them, thus building longer and longer expressions; recall that we recorded the students' screen (see Section 2.4.3 of Chapter 2) and were able to observe this behaviour.

The same analysis of tutor turns showed that the difference in tutors' language production between the two conditions was not significant. Interestingly, systematic and statistical differences were found when comparing student to tutor language behaviour; for instance, students' vs. tutors' NL/ME-token distributions. In both conditions, tutors used more natural language and fewer mathematical expressions than students. We did not analyse the the ME-length distributions further due to the previously-mentioned copy–paste artefacts.

From this point on we focus on a subset of the data: we look only at student utterances which contain natural language (NL and ME & NL categories in Tables 4.1 and 4.3). We start by looking at the distribution of utterance types.

4.3.3 Distribution of utterance types

The distribution of utterance types is shown in Table 4.6.[6] The majority of utterances are solution-contributing (74% in C-I, 67% in C-II), and most of them are proof steps. This is not surprising, of course. Proofs in the second experiment involved considering cases and proving both directions of a bi-conditional, which resulted in explicit verbalisations of proving strategy and proof structure, and in students signalling that a part of a proof is completed.

Among non-solution-contributing types, the largest class, 51%, are help requests: from general requests (‘‘Hilfe!’’ (*Help!*)) to specific requests, for

[6] Only the utterance types with more than five occurrences will be discussed here. Utterance types with lower frequency of occurrence are too sparse for any conclusions about their wording.

Table 4.6: Distribution of utterance types

	C-I	C-II	C-I&C-II
	Unique / Total	Unique / Total	Unique / Total
Solution-contributing	149 / 187	335 / 548	465 / 735
Proof contribution	143 / 180	326 / 539	450 / 719
Proof step	138 / 171	287 / 469	407 / 640
Proof strategy	4 / 4	25 / 30	29 / 34
Proof status	1 / 5	7 / 24	7 / 29
Proof structure	- / -	7 / 16	7 / 16
Meta-level	6 / 7	9 / 9	15 / 16
Self-evaluation	2 / 2	5 / 5	7 / 7
Restart	1 / 2	3 / 3	4 / 5
Give up	3 / 3	1 / 1	4 / 4
Other	46 / 64	193 / 267	231 / 331
Help request	16 / 16	136 / 154	149 / 170
Yes/No	1 / 18	1 / 24	1 / 42
Cognitive state	15 / 15	15 / 16	30 / 31
Politeness/Emotion/Attitude	2 / 3	14 / 21	14 / 24
Discourse marker	1 / 1	1 / 21	1 / 22
Answer	5 / 5	14 / 15	19 / 20
OK	1 / 1	1 / 6	1 / 7
Address	1 / 1	5 / 5	6 / 6
Session	- / -	4	4 / 4
Agree	2 / 2	1 / 1	3 / 3
Self-talk	2 / 2	- / -	2 / 2
Uninterpretable	3 / 3	4 / 4	7 / 7

instance, to provide a definition (‘‘Wie lautet die Definition der Operation ${}^{-1}$?’’ (*What’s the definition of* ${}^{-1}$*?*) or ‘‘Erklaere die Definition $R \circ S$ in Worten!’’ (*Explain the definition of $R \circ S$ in words!*)), or questions about propositions (such as ‘‘Ist (a, z) in R?’’ (*Is (a, z) in R?*) or ‘‘Elemente von $(R \circ S) \circ T$ sind Tripel der Form (x, y, z), oder?’’ (*Elements of $(R \circ S) \circ T$ are triples of the form (x, y, z), right?*)). The second largest category are closed-class types, Yes/No and OK, which make up 15% of non-solution-contributing utterances. The second largest open-ended verbalisations class are meta-cognitive statements on the state of knowledge (or, for the most part, *lack* thereof), 31 occurrences. Statements such as ‘‘Keine Ahnung mehr wie der Nachweis korrekt erbracht werden kann’’ (*No idea how the proof can be correctly produced*) or ‘‘Verstehe die definition nicht’’ (*Don’t understand the definition*), can be also interpreted as indirect requests for help. Only one wording appeared more than once, ‘‘Dann weiss ich nicht weiter’’ (*So I’m lost*).

Aside from two expressions of gratitude (‘‘Danke’’/‘‘Vielen Dank’’) and the four variants of apologies (‘‘Tut mir leid’’/‘‘Entschuldigung’’/

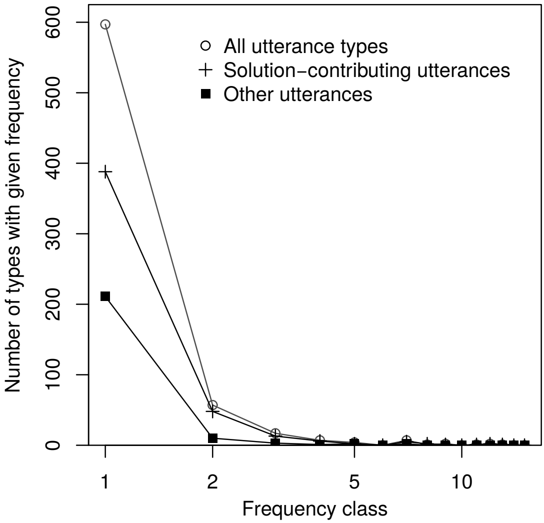


Figure 4.3: Frequency spectra: Utterance types (x-axis log-scaled)

''Verzeihung''/''Sorry''), the remaining expressions of emotions and attitude were idiosyncratic, spanning positive (''Das macht Spass mit Dir'' (*It's fun!*)) and negative polarity (''Wollen Sie mir nun Mathematik beibringen oder wollen Sie mich pruefen???'' (*Do you want to teach me math or is this a test???*), ''NERV!!'' (*[annoyance]*)). Not surprisingly, idiosyncratic were also other open-ended classes, *Answers* and *Addresses*, whose content is entirely determined be the preceding context (the tutor's turn which triggered them).

It is interesting that there were 22 occurrences of discourse markers and that they had a colloquial character, the kind typical of spoken language: ''na doll'', ''na ja'' (*oh well*), ''oh'', ''hm'', ''ach so'' (*oh, I see*), ''halt'' (*hang on*). The variety of discourse markers suggests that the subjects treated the dialogues much like spoken interaction, even though they were typewritten.

Figure 4.3 shows the frequency spectra of all the utterance types and the two major classes. The distribution of distinct verbalisations is heavily skewed. In all categories, the number of patterns occurring three to five times is less than 10. The tail of patterns with frequency 1 starts between 5-10 occurrences. In the ''All types'' set, the frequency-1 class covers 597 instances, whereas the remaining classes together 475 (44%). The frequency spectra also show that the data is sparse and even though some utterance types have a high frequency of occurrence (Table 4.6) they consist of mainly idiosyncratic linguistic patterns.

4.3.4 Proof contributions

Aside from the three classes of proof-level descriptions – proof strategy, structure, and status (see Table 4.2) – in the analysis that follows we distinguish three subclasses of proof steps. The subcategorisation takes into account *the type of content expressed in natural language* and *the type of linguistic knowledge which needs to be encoded in order for formalisation to be possible*.

The simplest case for translation are steps in which natural language is used only for logical operators (connectives and binders/quantifiers) or to signal proof step components, and where no discourse context nor domain-specific linguistic information is needed for interpretation. By proof step components we mean elements of a deduction system's proof language such as the declarative proof script language presented in (Autexier et al., 2012). In order to formalise proof steps of this kind, the only knowledge needed is that of the vocabulary and syntax of the natural language of logic (logical connectives) and of the proof structural markers (proof discourse connectives); that is, only a basic interpretation lexicon. Examples of this class of verbalisations include:

Wenn $A \subseteq K(B)$, dann $A \cap B = \emptyset$
If $A \subseteq K(B)$, then $A \cap B = \emptyset$

Sei $(a, b) \in (R \circ S)^{-1}$
Let $(a, b) \in (R \circ S)^{-1}$

We will refer to this class as *Logic & proof step components*.

The second and third class of verbalisations are those which require contextual and domain knowledge for interpretation and formalisation. If beyond the type of content described above, only domain concepts (here: from set theory and binary relations) and discourse references have to be translated, then the proof step belongs to the category *Domain & context*. The domain concepts may be named using single or multi-word terms or using informal wording, such as the locative prepositional phrase with ''in'' to stand for the set membership relation. Examples of the second class of proof steps include:

in $K(B)$ sind alle x, die nicht in B sind
in $K(B)$ are all x which are not in B

Nach der Definition von $\circ$ folgt dann (a, b) ist in $S^{-1} \circ R^{-1}$
By definition of $\circ$ it follows then that (a, b) is in $S^{-1} \circ R^{-1}$

Nach Aufgabe A gilt $(R \cup S) \circ T = (R \circ T) \cup (S \circ T)$
By Exercise A it holds that $(R \cup S) \circ T = (R \circ T) \cup (S \circ T)$

In the last example, the reference ''Aufgabe A'' (*Exercise A*) needs to be resolved. Note that the utterance ''Es gilt nach Definition ausserdem

$S^{-1} \circ R^{-1}$ =...’’ (*By definition it moreover holds that...*) still belongs to the class *Logic & proof step components* because no domain-specific vocabulary is used; the word ‘‘definition’’ is in the basic lexicon of mathematics and ‘‘by definition’’ expresses justification in general.

The third class comprises steps which are not spelled out explicitly, but rather as high-level meta-descriptions of a (possibly complex) transformation which needs to be performed. An example of such a descriptive step is C-II S8 in Figure 4.1: *The proof goes exactly as above since in step 2 to 6 there are only equivalences*. Other examples include:

> Analog geht der Fall, wenn $(a, z) \in S$
> *The case for $(a, z) \in S$ is analogous*
>
> de morgan regel 2 auf beide komplemente angewendet
> *de morgan rule 2 applied to both complements*
>
> $(S \circ T)$ ist genauso definiert
> *$(S \circ T)$ is defined the same way*

Complex proof steps of this kind will be referred to as *Meta-level description*. The three subclasses of *Proof contributions* are summarised below:

Logic & proof step components	Only logical connectives and components of a proof step need to be interpreted.
Domain & context	Domain terminology and contextual references need to be interpreted (as well as, possibly, logical connectives and proof step components).
Meta-level description	An indirect proof step specification needs to be interpreted (as well as, possibly, all of the above).

An alternative classification, designed with a motivation similar to ours, has been proposed by Wagner and Lesourd (2008). It is also verbalisation-oriented, however, it is imprecise. First, it is not clear whether the class *simple connections* accommodates utterances with adverbs or adverbial phrases, such as ‘‘Moreover, as previously shown, it follows that...’’ Second, and more importantly, the distinction between *weakly verbalised* and *strongly verbalised* formulas is unclear. *Weakly verbalised* formulas are defined as those ‘‘where some relations or quantifiers are partly verbalised’’, while *strongly verbalised* formulas as those ‘‘where all relations and quantifiers are fully verbalised’’. Based on these definitions it is not clear why the example ‘‘a is the limit of $(a_n)_{n \in N}$’’, given in the paper, should be classified as *weakly verbalised*, whereas ‘‘For all ϵ holds: there exists a $n_0(\epsilon) \in N$ with...’’ as *strongly verbalised*; clearly, the set membership relation in $n_0(\epsilon) \in N$ is not verbalised.

Table 4.7: Descriptive information on proof contributions

	C-I	C-II	C-I&C-II
	Unique / Total	Unique / Total	Unique / Total
Proof step	138/ 171	287 / 469	407 / 640
Logic & proof step components	54 / 80	136 / 286	175 / 366
Domain & context	78 / 85	140 / 171	216 / 256
Meta-level description	6 / 6	11 / 12	16 / 18
Proof strategy	4 / 4	25 / 30	29 / 34
Proof structure	- / -	7 / 16	7 / 16
Proof status	1 / 5	7 / 24	7 / 29

Table 4.7 shows descriptive statistics on proof contributions, with proof steps subclassified as described above. Not surprisingly, the wording of proof contributions which refer to proof-level concepts – proof strategy and proof structure – is diverse. Wording of proof status utterances is repetitive; indeed, most often only the end of the proof is signalled explicitly and most often using the abbreviation ‘‘q.e.d.’’[7] Now, also not surprisingly, within the class of proof steps, the more complex the content, the more varied the wording. Meta-level descriptions of proofs are almost entirely idiosyncratic. Only two utterance patterns occurred more than once: ‘‘MATHEXPR ist analog definiert’’ (*MATHEXPR is defined analogously*) and ‘‘das gleiche gilt fuer MATHEXPR’’ (*The same holds for MATHEXPR*). Wording in the *Domain & context* category is also diverse: 92% of instances are distinct in C-I, 82% in C-II, and 84% overall. Most repetitive patterns are found in the *Logic & proof step components* class: 67% of all utterance instances in this category are distinct in C-I, only 47% in C-II, and 48% in both corpora combined. Overall, 63% of proof step verbalisations (from all the three categories) are distinct.

Figure 4.4 shows the frequency spectra of the three proof step categories in the combined corpus, C-I&C-II. Again, the distribution is heavily skewed. In the largest category, *Domain & context*, 210 out of 216 unique patterns occur only once or twice; that is 97% (191 patterns occur once; 75% of all instances in this category). In the *Logic & proof step components* category, around 150 out of the 175 unique patterns, 73%, occur once or twice, and there are only 8 patterns with at least five instances of occurrence (128 patterns occur once, 35% of instances in this category). Table 4.8 shows the top-10 most frequent linguistic patterns in the three classes of proof steps from the combined corpus, C-I&C-II, with their frequency of occurrence. Recall, moreover, that only 20 solution-contributing utterances occurred in both corpora (see Table 4.2).

[7] Recall that the different spelling and verbalisation variants of ‘‘q.e.d.’’ have been normalised.

Table 4.8: Top-10 most frequent utterance patterns expressing proof steps

Type	Linguistic pattern	Frequency
Logic & proof step components	sei MATHEXPR	54
	es gilt MATHEXPR	13
	wenn MATHEXPR dann MATHEXPR	12
	also MATHEXPR	12
	dann ist MATHEXPR	11
	also ist MATHEXPR	9
	MATHEXPR und MATHEXPR	8
	MATHEXPR ist dann MATHEXPR	7
	daraus folgt MATHEXPR	7
	daraus folgt dass MATHEXPR	7
Domain & context	nach REFERENCE MATHEXPR	7
	DOMAINTERM	7
	nach REFERENCE ist MATHEXPR	4
	MATHEXPR nach REFERENCE	3
	DOMAINTERM von MATHEXPR ist DOMAINTERM MATHEXPR	3
	aus REFERENCE folgt MATHEXPR	3
	wegen der formel fuer DOMAINTERM folgt MATHEXPR	2
	oder MATHEXPR wegen DOMAINTERM von MATHEXPR	2
	nach REFERENCE gilt MATHEXPR	2
	nach DOMAINTERM gibt es ein MATHEXPR mit MATHEXPR	2
Meta-level description	MATHEXPR ist analog definiert	2
	das gleiche gilt fuer MATHEXPR	2
	gleiches gilt mit MATHEXPR	1
	DOMAINTERM auf beide DOMAINTERM angewendet	1
	der fall MATHEXPR verlaeuft analog	1
	der beweis von MATHEXPR ist analog zum beweis von MATHEXPR	1
	beweis geht genauso wie oben da in REFERENCE bis REFERENCE nur DOMAINTERM umformungen stattfinden	1
	analog geht der fall wenn MATHEXPR	1
	andersrum	1
	die zweite DOMAINTERM ergibt sich aus der umkehrung aller bisherigen beweisschritte	1

4.3.5 Growth of the diversity of forms

Finally, we are interested in how the diversity of forms evolves with an increasing number of dialogues. Specifically, we would like to know how many dialogues are needed to have observed most of the verbalisation patterns.

Figure 4.5 (p. 172) shows a plot of a variant of the type–token (vocabulary growth) curve (Youmans, 1990). Verbalisation patterns are used as vocabulary. On the x-axis is the number of dialogues seen. Rather than the raw type count, the y-axis shows the proportion of observed pattern types out of all pattern

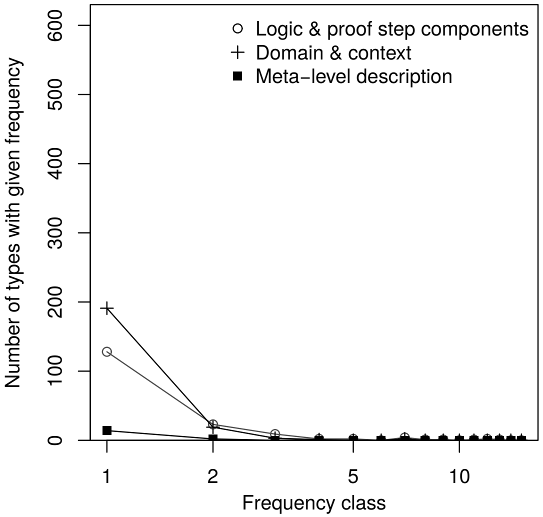


Figure 4.4: Frequency spectra: Proof step types (x-axis log-scaled; y-axis range extended to match Figure 4.3 for comparison)

types in the given corpus.[8] For the C-I&C-II plot, the corpora were combined and 10 random dialogue sequences were drawn from the combined set.

What can be seen from the graphs is that the pattern vocabulary grows linearly, showing, however, a large variance over the 10 samples drawn, especially in the combined data set. The tendency is similar in both corpora: on average, half of the patterns have been seen at about 40% of the data sets and 80% of the patterns at about 75-80% of the data set in C-I (ca. 17 dialogues) and 70-75% in C-II (ca. 26 dialogues). In the combined corpus, however, depending on the sample drawn, half of the patterns can have been seen already about 30-40% into the data set. Likewise, around 80% of the patterns, for some samples, have been seen about 65-70% into the data set (ca. 35-40 dialogues).

4.4 Conclusions

The results show that the language of students' proofs is not as repetitive as one might expect. Students produce complex utterances during meta-communication and when contributing proof steps. 57% of utterances in C-I and 73% in C-II contained natural language. More natural language in C-II

[8] 198 NL + ME & NL patterns in C-I, 530 in C-II, and 700 in C-I&C-II; see Table 4.1 (p. 161).

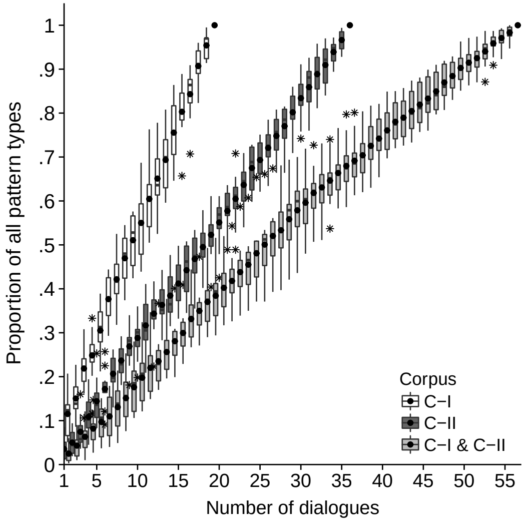


Figure 4.5: Growth of verbalisation patterns over 10 random dialogue sequences

may be due to the higher complexity of binary relations proofs. However, set theory is very naturally expressed in natural language, so gaining insight into why this was the case requires further investigation.

An analysis of the C-II data revealed differences in language production between the two study material conditions. The VM-group tended to use more natural language than the FM-group and subjects' dialogue turns contained more, but shorter, mathematical expressions. The FM-group tended to use more and longer formulas overall, and less natural language. Since there was no significant difference between tutors' dialogue behaviour with respect to language production between conditions, the differences in dialogue styles must have been at least partly due to the study material format having a priming-like effect. Another factor that may have contributed to the differences could involve students' individual differences in mathematical skills or specific dialogue styles of subject–wizard pairs having to do with the student's skills.

The results on the influence of the study material presentation have implications for the implementation of tutorial dialogue systems. On the one hand, more natural language, be it resulting from a verbose presentation of the material or from the students' individual preference for a particular style, imposes more challenges on the input understanding component. In the context of mathematics, this involves a reliable, robust parser and discourse analyser capable of interpreting the mixed language. On the other hand, prompting for more symbolic language by presenting formalised material imposes stronger requirements on the mathematical expression parser since longer expressions tend to be prone to errors. The same holds of the copy–paste functionality: while convenient for the user, it may lead to sloppiness in revising the copied text. This, in turn, calls for flexible formula parsing, error correction (such as the one we present in Section 6.4), and dialogue strategies addressing formulas with errors (such as those we proposed in (Horacek and Wolska, 2007, 2008)).

From the pedagogical point of view, the study material format should be adequate to the tutoring goals: in teaching formal proofs more rigour should be imposed than in informal proofs. The material should be also adapted to the skills of the student: formal presentation may lead to an inefficient dialogue with a novice, centering around such issues as syntactic formalities, instead of the higher-level goal of teaching problem solving (recall the discussion in Chapter 2). The general issue arising here is what study material format a tutoring system should present to the student. An advantage of verbose material, including worded explanations, is that a novice can compensate for lack of familiarity with formal notation and still attempt to construct proofs. Advanced students may be able to express proofs formally anyhow, while the verbosity of the material might encourage them to produce conceptual sketches of proofs typical of skilled mathematicians. This assumes that the tutoring system's interpretation and dialogue management modules can handle a variety of discourse and dialogue phenomena, including telegraphic fragmentary utterances and informal descriptions (discussed in Section 3.2.2).

The wording of proof steps is surprisingly diverse and the language in the two corpora different. Among the 28 verbalisation patterns common to both corpora there were 20 proof steps, of that the majority in the *Logic & proof step components* type. The low number of common patterns is reflected in the type–token plot (Figure 4.5) which exhibits a steady increase with only one area of slower growth in the combined corpus, about 20-25% into the data set. The difference in the linguistic diversity of the proof language (the proof contributions class) in the two corpora can be also seen in the different distributions of distinct linguistic patterns (Table 4.7). In the *Logic & proof step components* class, 67% of the verbalisations were distinct in C-I and 47% in C-II. In the *Domain & context* class, 92% of all the verbalisations were distinct in C-I and 82% in C-II. That is, the language in C-II appears more repetitive. In both corpora, however, the language in the latter class is more heterogeneous than in the former.[9] The frequency spectra and the pattern growth curves show further the degree to which the language is indeed diverse.

[9] The *Meta-level descriptions* are too sparse to draw conclusions (18 occurrences overall).

In the *Logic & proof step components* class, around 75% of the distinct types were single-occurrence utterances. In the *Domain & context* class, around 90% of the types were single-occurrence.

Not surprisingly, the majority of the meta-level communication are students' requests for assistance: requests for hints, definitions, explanations, etc. Out of the 170 help requests, 149 (88%) were distinct verbalisations; 136 single-occurrence patterns. A further subclassification of help requests might reveal more homogeneity in the wording within subcategories. The relatively large number of discourse markers typical of spoken interaction suggests that participants had an informal approach to dialogue and treated it like a chat, adapting spoken language, which they would have used in a natural setting, to the experiments' typewritten modality. This is a known phenomenon (Hård af Segerstad, 2002). The diversity of verbalisations may be partly due to this.

The key conclusion which can be drawn from the analyses is that in a tutoring setting, even the seemingly linguistically predictable domain of mathematical proofs is characterised by a large variety of linguistic patterns of expression, a large number of idiosyncratic verbalisations, and that the meta-communicative part of discourse has a conversational character, suggesting the students' informal attitude towards computer-based dialogues and their high expectations of the input interpretation resources. This calls for a combination of shallow and deep semantic processing: shallow pattern-based methods for contributions which do not add to the proof and semantic grammars for proof-relevant content, in order to optimise coverage. In the next chapter we propose a language processing architecture for analysing students' proof language. In Chapter 7, we show that deep lexicalised grammars for parsing proof contributions provide better generalisation and thus better scalability in terms of coverage than a phrase-based formalism.

Chapter 5

Processing informal mathematical discourse

In this chapter we describe an architecture for processing informal mathematical discourse. We start by motivating the general properties of the architecture and of the interpretation strategy which we propose. Then we present the high-level interpretation processes, discuss their components and the employed methods of language analysis. The presentation of the interpretation strategy for mathematical discourse is divided into two parts: In Section 5.2.3, we present the basic approach to processing mathematical language motivated by the most prominent language phenomena discussed in Chapter 3 and in Section 5.3 we show a complete walk-through analysis of an example utterance from the corpus. In the following chapter, Chapter 6, we show how we model selected language phenomena found in our corpora in more detail and discuss various extensions to the basic resources for processing a subset of the language phenomena. Material presented in this chapter appeared in (Wolska and Kruijff-Korbayová, 2004a; Wolska et al., 2010).

5.1 Rationale of the approach

The approach to mathematical discourse processing which we adopt rests on a number of well-motivated design principles: The underlying philosophy of our approach is *modularity*, that is, encapsulation of information required for the different processing tasks and of the processes themselves, and *parameterisation*. We argue that in order to be able to address the peculiarity of mathematical discourse, that of fluently interleaving natural language and mathematical notation (discussed in Chapter 3), the interpretation strategy for mathematical language should be such that the information contributed by the two language modes can be seamlessly integrated into the semantics of utterances presented in the mixed language. We propose to achieve this by means of *encapsulation of symbolic content* and *a uniform processing strategy*, the same for utterances presented in natural language as well as those presented in mixed language. Considering the complexity of the language phenomena and the fact that we are aiming at a uniform analysis of language phenomena of various complexity, we argue for *deep linguistic analysis* as a method of providing a systematic and consistent account of the mixed-language discourse and propose a *stepwise interpretation process* in which a representation of the utterance's semantics is gradually enriched with more specific information. Finally, we argue that the output representation produced by the language processing component should not be specific to a proof representation language of a particular deduction system. It should be rather a *linguistically-motivated output representation, independent of a domain reasoner*. In the following sections we briefly elaborate on the motivation behind these design decisions.

Modularity and parameterisation Modularity in complex systems is a desirable feature as such because, among other reasons, rigorous definition of the modules' interfaces facilitates exchange of processing methods. In language processing, modularity is a natural choice because the individual linguistic processing tasks are structurally and functionally different. In the case of mathematical discourse, it is also motivated by the fact the specification of the processes of certain architecture components needs to be parameterised with respect to a number of variables (which we will discuss in Section 5.2.1) in order to facilitate portability across scenarios. First, at the level of the larger architecture, the linguistic analysis (which operates on language input) and domain reasoning (which operates on constructed symbolic representations of proof contributions) need to be clearly separated (see Figure 1.2, p. 37). Second, the architecture encapsulates language processing subcomponents which process input in a stepwise fashion, contributing information at different levels of granularity of linguistic analysis. Thus, similarly to Zinn's (2006) and mArachna's (Jeschke et al., 2008) approaches, we argue for a highly modular architecture for processing mathematical discourse. However, our architecture includes components whose processes are functionally self-contained and which the other approaches integrate into larger components (for instance, mathematical notation processing) or do not mention at all (for instance, parsing mixed language or interpretation of imprecise wording).

Encapsulation of mathematical expressions and uniform processing As illustrated in Section 3.2.2.1 (p. 112), mathematical notation can be seamlessly embedded into natural language. While in certain contexts, the presence of symbolic expressions may be a source of deviation from the norms of natural language syntax,[1] symbolic expressions behave just like other linguistic entities in that they enter into grammatical and semantic relations with other constituents in a sentence (or dialogue utterance). Therefore, we propose to treat mathematical notation constituents the same way as linguistic content, while abstracting from the individual symbols of which they are composed. In other words, we argue for uniform processing of the two language modes at the level of utterance or sentence syntax in which meaningful constituents of mathematical expressions *as wholes*, rather than symbols individually, are treated as tokens by the natural language parser.

The interpretation process we propose comprises a number of steps during which mathematical expressions are first encapsulated and subsequently analysed as structured linguistic constituents represented as special lexical or clausal units (‘‘pseudo-lexemes’’) in the parser’s grammar. In the course of parsing, content encapsulated in this way is treated on a par with natural language lexical units. This approach is superior to that of representing individual symbols of mathematical notation within the parser’s lexicon, as proposed by Zinn (see (Zinn, 2004, Section 5.2)) because it supports modularity and parameterisation: parsing mathematical expressions can be delegated to a mathematical notation parser which has access to its own resources and parsing knowledge adapted to the notation format and mathematical domain in question. Clearly, it is also superior to mArachna’s approach in which mathematical notation within sentences is not at all analysed in the context of the natural language within which it is embedded (see (Jeschke et al., 2008)), which obviously results in information loss.[2] More details and example analyses will be presented in Sections 5.2.2, 5.2.3, and in Chapter 6.

Deep linguistic analysis Traditionally, two approaches to language processing are distinguished in computational linguistics: ‘‘shallow processing’’ typically refers to approaches based on more or less coarse-grained lexico-syntactic information, such as information on word classes (parts of speech), phrase (noun phrases, verb phrases, predicate–argument structures) and clause structure, or statistical word cooccurrence information, but without or with only limited access to semantics. Information and document retrieval is usually performed based on this kind of ‘‘shallow’’ information. At the other end of the spectrum is ‘‘deep processing’’ which uses semantic parsers to construct symbolic representations of (possibly underspecified) semantics, so-called *logical*

[1] Non-standard syntax is, however, characteristic of sublanguages of which mathematical language is an example. We discussed these phenomena in Sections 3.1.1 and 3.2.2.3.

[2] The same approach, proposed originally in (Wolska and Kruijff-Korbayová, 2004a), has been also adopted in LeActiveMath project (Callaway et al., 2006).

form, based on the sentence's surface form. Logical forms represent context-independent (literal) meaning of sentences (utterances) and they are typically based on a logic notation, such as the Montagovian (simply-typed) lambda calculus or other quantified or quantifier-free languages (see, for instance, (Alshawi and Crouch, 1992; Copestake et al., 1995)). Shallow processing offers robustness – while the result of processing may not always be correct, *a* result is always produced – however, while it is possible to produce some semantic representation based on shallow processing, the representation may be incomplete.[3] Considering the fact that we aim at a formal representation which can be reliably mapped to an input language of a deduction system, we argue for a deep processing approach for mathematical discourse.

The advantage of a deep approach is that the syntax–semantics interface, that is, the mapping of lexico-syntactic forms to logical forms, is well-defined and ensures precision in meaning assignment thanks to this explicit definition. Semantic representations are derived in a principled way based on the notion of *compositionality of meaning*.[4] Deep semantic parsers use carefully hand-crafted grammars, typically *lexicalised grammars*, which encode language phenomena based on principled linguistic analysis. Examples of such formalisms are Head-driven Phrase Structure Grammar (Pollard and Sag, 1994), Lexical-Functional Grammar (LFG) (Bresnan, 2001), or Categorial Grammar (CG) (Ajdukiewicz, 1935; Bar-Hillel, 1953; Lambek, 1958).

As the core of our processing architecture we propose Combinatory Categorial Grammar (CCG) and a dependency-based semantic representation. CCG is a variant of categorial grammar in which categories (categorial grammar types) associated with lexemes are combined using a set of rules (Steedman, 2000). The specific ''multi-modal'' variant of CCG and its implementation, which we adopt, provide a way of controlling derivations by restricting rule application by means of features on categories and modes on category-building operators (more in Section 5.2.3.1). These mechanisms are particularly relevant when modelling languages with relatively free word order, such as German. Our semantic representations, produced in parallel with syntactic derivations, are based on the Praguian notion of *tectogrammatics* and reflect the *semantic dependency structure* of the parsed sentences (Sgall et al., 1986). Semantics

[3] A variety of language processing architectures can be described as *hybrid approaches*, that is, approaches which either use both shallow and deep methods for processing language or attempt to integrate various processing components. Heart of Gold (Schäfer, 2006) is an example of a hybrid system in which such an integration is done in a principled way using (R)MRS as semantic representations (Copestake et al., 2005).

[4] The notion of ''logical form'' goes back to the work of Tarski, Russell, and Frege. The ''Principle of compositionality'' is due to Frege. Work on ''translation'' of natural language into logic dates back to the early 70s and the work of Montague, Partee, Dowty, May, and Cooper, among others.

in this sense is context-independent and models the *literal meaning* of the utterances. Thus, our basic formalisation of the language of mathematics is in terms of the *linguistic meaning* of mathematical content, modelled as semantic dependency structures. These structures are formally represented using Hybrid Logic Dependency Semantics (HLDS) (Baldridge and Kruijff, 2002), a semantic formalism based on the syntax of hybrid modal logic (Blackburn, 2000). Linguistic meaning is subsequently interpreted in the context of mathematical domain and the semantic representation is enriched with domain-specific information (see below). Details on parsing and further processing of the dependency structures will follow in Section 5.2.3 of this chapter and Section 6.1 of the next chapter. An illustration of semantic interpretation based on transforming dependency structures will be shown in Section 6.2.2 when discussing the interpretation of the ‘‘the other way round’’ operator.

Stepwise interpretation Similarly to many other language processing systems, the architecture we propose for processing mathematical language is based on a sequence of analysis steps which attempt to provide gradually more specific information about the input under analysis. Once high-level information on the structure of a communicative unit is known (that is, information on the utterance units’ boundaries and the boundaries of symbolic mathematical expressions within the utterance units) meaning assignment starts with semantic parsing (briefly outlined above). At this stage, our basic semantic representation is *domain-independent* and represents the linguistic meaning of an utterance under consideration in terms of a dependency structure. Subsequent analysis steps operate on this representation attempting to assign a more precise, *domain-specific*, interpretation to its elements. These subsequent interpretation processes enrich the original semantic representation with further information if it can be found based on dedicated resources: a semantic lexicon and a linguistically-motivated domain model. The resulting output representation can be thought of as an *interpreted dependency structure*. The interpretation process will be further elaborated in Section 5.2.3.2, while more details on the structure of the interpretation resources will be presented in Section 6.2.1.

Linguistically-motivated reasoner-independent output representation In the architecture for processing mathematical discourse which we envisage – recall Figure 1.2 (p. 37) – domain reasoning and language processing tasks are clearly separated. The reason for this is that a generic language interpretation component does not have knowledge to reason about discourse at the domain level; that is, reason about the proofs. *Linguistic* analysis is what it is: it is an analysis of the *language* itself. While certain inferences can be made based solely on the verbally expressed content (for instance, sortal restrictions violations), many domain-specific mathematical inferences cannot. For instance, it is impossible to decide on the scope of a sentence-initial discourse marker ‘‘hence’’, which introduces a conclusion from *one or more* previously stated proof steps, without the knowledge of the logical structure of the proof. Therefore, we argue that the core linguistic analysis in a system for processing proofs may stop short of any interpretation which requires knowledge of mathematics beyond the knowledge of the *language* used to talk about mathematics. The representation itself should be *linguistic*, rather than express the communicated mathematical content directly in a formal language of logic or of a specific deduction system. On the contrary: in order to facilitate portability, the output representation of the language interpretation process should not be specific to any deduction system. HLDS-based interpreted semantic dependency representations have this property. Translation of the semantic representations into an input language of a domain reasoner should be performed by the proof representation processing component – see Section 1.2 – as this translation is entirely reasoner-specific, that is, dependent on the input language of the deduction system employed for domain reasoning tasks.

In the following sections, we present a modular architecture for processing informal mathematical discourse designed according to the principles discussed above. We first introduce the core components of the architecture and then elaborate on our approach to computational interpretation of informal proof discourse in the scenarios introduced in Chapter 1. The presentation of the interpretation strategy proper is divided into two parts: First we present the basic analysis steps which address a set of simple, but frequent linguistic phenomena and illustrate the analysis process with a walk-through example. Methods of modelling specific selected phenomena in students' language are presented in Chapter 6.

5.2 Language processing architecture

The language processing architecture we propose for mathematical discourse is built on a pipeline of (standard) larger language processing subcomponents: preprocessing, parsing, and sentence- and discourse-level interpretation. Their design and functionality is motivated by the properties of mathematical language, discussed in Chapter 3. The overall architecture is shown in Figure 5.1. In the reminder of this chapter we present the individual processing components and their functionality, including the core contribution of this thesis: an interpretation strategy for the language of mathematical proofs. Details on how specific

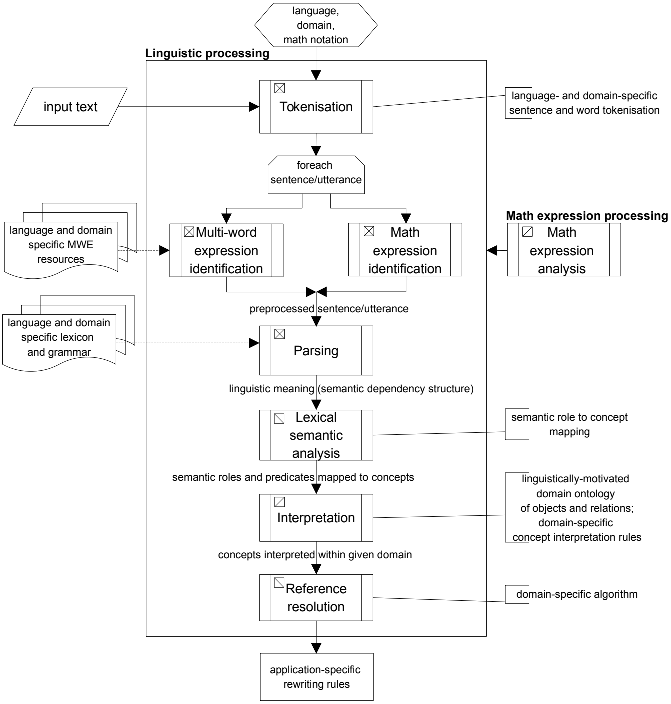

language,
domain,
math notation
Linguistic processing
input text
☒ Tokenisation
language- and domain-specific sentence and word tokenisation
foreach sentence/utterance
language and domain specific MWE resources
☒ Multi-word expression identification
☒ Math expression identification
Math expression processing
◩ Math expression analysis
preprocessed sentence/utterance
language and domain specific lexicon and grammar
☒ Parsing
linguistic meaning (semantic dependency structure)
◩ Lexical semantic analysis
semantic role to concept mapping
semantic roles and predicates mapped to concepts
◩ Interpretation
linguistically-motivated domain ontology of objects and relations; domain-specific concept interpretation rules
concepts interpreted within given domain
◩ Reference resolution
domain-specific algorithm
application-specific rewriting rules

Figure 5.1: Architecture for processing informal mathematical language

language phenomena are processed will be presented in the next chapter. When discussing the interpretation strategy we focus on proof contributions and do not address other types of communicative units. Non-solution-contributing utterances would lend themselves better to shallow processing methods since, first, they do not need a translation to formal language, and second, due to the variety in their verbalisations (discussed in Section 4.3.3). We start by introducing three obvious variables with respect to which a larger system for processing mathematical language must be parameterised.

5.2.1 Parameters

In order to facilitate portability across scenarios, the language processing system is parameterised with respect to the following three variables:

- the natural language of the contributions,
- the mathematical domain, and
- the format of the mathematical notation.

Parameterisation with respect to the input language is a obvious: parsing is language-specific, hence the architecture’s input analyser should support grammars, or language models in general, of different natural languages in which proof contributions can be expressed. The language models, in turn, should comprise appropriate terminological lexica for the mathematical subarea of the given discourse. (These are also dependent on the mathematical domain of the proof discourse under analysis.) Before syntactic and semantic analysis can proceed, preprocessing modules prepare the input for parsing by identifying utterances, (multi-)word units, and elements of mathematical notation within the input communicative units. Sentence (or utterance) and word boundary detection by themselves are language specific. The process of identification and analysis of mathematical notation, however, must be specialised both with respect to a mathematical domain (the set of symbols used and their semantics differ across domains; recall the discussion in Section 3.2.1) and also with respect to the natural language of the input (in English, for instance, the token ‘‘a’’ needs to be disambiguated between an indefinite article and a mathematical symbol). Identification of symbolic mathematical expressions within natural language needs to be moreover parameterised with respect to the input format in which mathematical expressions are entered. LaTeX (Knuth, 1986) is a de facto standard for mathematical document formatting for scientific publications. While the document processing scenario would most likely involve LaTeX-based documents, possibly further processed using a dedicated mathematical document processing system, such as LaTeXML (Stamerjohanns et al., 2010), tutoring environments and web-based interactive proof checkers would typically offer a graphical user interface with buttons for entering mathematical symbols. In this case, the underlying representation format for mathematical expressions might be MathML or OpenMath[5] or, as was the case with our corpora, a custom format for representing mathematical symbols as ASCII text, for instance, for the purpose of storing interaction logs. In Figure 5.1 (p. 181) the components marked with ⧅ in the top left corner are those whose resources are specific to the natural language, ⧄ marks dependency on mathematical domain, and ☒ marks processing which depends on both the language and the mathematical domain.

5.2.2 Preprocessing

By ‘‘preprocessing’’ in language technology one understands the part of text processing whose purpose is to prepare the input for the analysis proper. Typical preprocessing steps include sentence and word boundary detection

[5] http://www.w3.org/MathML, http://www.openmath.org

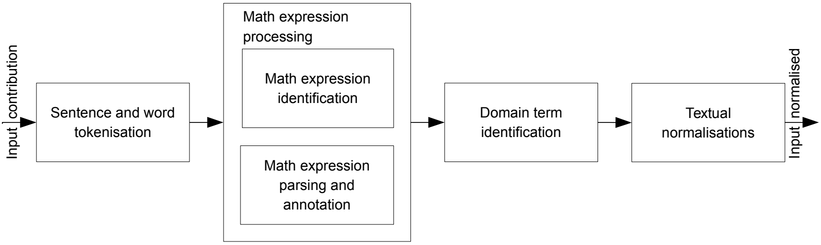

Input contribution
Sentence and word tokenisation
Math expression processing
Math expression identification
Math expression parsing and annotation
Domain term identification
Textual normalisations
Input normalised

Figure 5.2: Preprocessing

(or tokenisation), simple stemming or full morphological analysis, part of speech tagging, that is, identifying a lexeme's word class, etc.

Our parsing process is based on a lexicalised grammar, that is, all word forms as well as their word classes are explicitly specified in the parser's lexicon. Therefore, in the present architecture, input is not stemmed nor part of speech tagged. However, preprocessing mathematical discourse, as well as any type of technical discourse which uses mathematics as its formal language, aside from the standard sentence and word tokenisation, involves identifying and analysing symbolic mathematical expressions as well as identifying domain terms, the technical vocabulary of the special language. Figure 5.2 shows a general preprocessing pipeline for mathematical discourse. The three preprocessing steps are outlined in the following sections. For the low-level ‘‘normalisation’’ step, please refer to Section 4.2.3 (p. 157).

5.2.2.1 Sentence and word tokenisation

The purpose of the tokenisation process is to segment the input contributions into utterances (or possibly sentences) and word-like units (tokens), that is, identify utterance and word boundaries. As we have pointed out before, sentence and word boundary detection is language specific. Moreover, in order to account for the symbolic expressions embedded within the natural language text, the process distinguishes between tokens which are natural language lexemes and those which form part of symbolic mathematical expressions.

Although conceptually simple, in general, automatic sentence and word tokenisation are non-trivial tasks; see (Grefenstette and Tapanainen, 1994) for a discussion of tokenisation issues. Approaches to sentence boundary detection in narrative text range from simple heuristics to statistical, machine learning approaches; see, for instance, (Reynar and Ratnaparkhi, 1997; Palmer and Hearst, 1997; Mikheev, 2000; Silla Jr. and Kaestner, 2004; Kiss and Strunk, 2006). In dialogue-based interaction input may be ill-formed, in particular, punctuation may be omitted. In the two collected corpora, 40% of utterances either lacked the final punctuation or the utterance final punctuation was non-standard, for instance, a comma or colon were used, as in (59) and (60):

(59) Dann ist $(A \cup C) = A$, und $(B \cup C) = B\boxed{,}$ daraus folgt der Beweis, $A \cap B \in P(A \cap B)$

Then $(A \cup C) = A$, and $(B \cup C) = B$, the proof follows from this, . . .

(60) das wars: wenn $A \subseteq K(B)$, dann sind A und B verschieden, haben keine gemeinsamen Elemente$\boxed{,}$ daraus folgt, dass $B \subseteq K(A)$ sein muss

that's it: if $A \subseteq K(B)$, then A and B are different, have no common elements, it follows from that that $B \subseteq K(A)$ must hold

Since tokenisation issues are not the main focus of this work, we implemented only simple procedures for the tokenisation step of preprocessing, which, however, ensured that our entire data set is correctly processed. Sentence and word tokenisation of both corpora has been performed using a set of regular expressions, as in the method proposed by Grefenstette and Tapanainen. Sentence and word tokenisers were iteratively tuned in such way that both corpora have been correctly processed, that is, we adjusted and extended the regular expressions, reprocessed the data, and verified the accuracy by inspecting the results, until the corpora were processed without errors. For the purpose of the evaluation presented in Chapter 7, utterances have been manually segmented as described in Chapter 4. Since we focus on semantic analysis, we do not address the tokenisation step any further in this thesis.

5.2.2.2 Domain term identification

Mathematics, a specialised domain, is rich in technical vocabulary: domain terms which name objects about which mathematical discourse treats. Clearly, an architecture for processing mathematical discourse needs to be capable of identifying and interpreting mathematical terminology. Examples of technical vocabulary from both of our corpora as well as other mathematical subareas, both single and multi-word units, were presented in Section 3.2.2.2 (p. 115). Because our experiments were set in only two mathematical domains, set theory and binary relations, and covered only small subsets of those domains, the set of technical terms appearing in the corpora is not large: there are 111 instances of nominal (noun phrase) domain terms in the set theory corpus and 250 instances of nominal domain terms in the binary relations corpus.

Terminology identification and extraction as well as identification of multiword expressions are research subareas in their own right. The currently prevalent approaches to domain term and multi-word unit identification are based on corpus statistics and machine learning; examples of recent work include (Frantzi et al., 2000; Pazienza et al., 2005) or (Kubo et al., 2010). Given the restricted scope of our experiments we employed a simple lexicon-based approach to identifying domain terms. For the purpose of the analyses in Chapter 4 and the evaluation in Chapter 7, domain terms have been identified based on a list extracted from the collected corpora and the background reading material. The list included all wording variants of single- and multi-word *nominal* units (examples of different wording variants of de Morgan's Laws and Distributivity of Union over Intersection have been shown in Section 3.2.2.2; p. 115). In order to account for misspellings and inflectional suffixes, a simple fuzzy matching procedure based on string edit distance (Levenshtein) has been implemented. Output of the domain term tagger has been verified and corrected manually. Since in this thesis we do not focus on domain term identification as such, we ascertained that the terminology lists are exhaustive for the collected corpora. We do not address the domain term identification process any further. However, important from the point of view of the interpretation strategy is how domain terms are treated during processing.

In the approach we propose, nominal single- and multi-word domain terms, once identified, are abstracted over in the course of syntactic and semantic parsing. The meaning of domain terms is incorporated into semantic representations at the interpretation stage, following semantic parsing. In practice, as part of preprocessing, we substitute each occurrence of a domain term with a symbolic token which represents it; as described in Section 4.2.3. This can be considered a kind of textual normalisation step. In our implementation the string DOMAINTERM was used to represent technical terminology. We argue that this approach is well-motivated and adequate for mathematical discourse for two reasons: First, once a lexical unit is identified as a domain term, its interpretation requires also domain knowledge and not just the sentence context. (Recall the ‘‘left ideal’’ example from Section 3.2.2.4 of Chapter 3.) Second, separating the two analysis processes enables better resource management. The parsing lexicon becomes smaller and focused on sentence-level phenomena, while domain terms can be handled by a dedicated noun phrase grammar with a terminological lexicon comprising solely noun phrase forming word classes: articles, adjectives, participles, nouns, and prepositions.

5.2.2.3 Processing mathematical expressions

Unlike typical genres which are commonly addressed in natural language processing, for instance, news text or general narrative prose, mathematical discourse requires that the symbolic mathematical expressions, mathematical notation which forms an inherent part of content, be interpreted in the context of the natural language within which they are embedded. To date, large scale efforts at processing scientific discourse tend to address higher level tasks (for instance, argumentative structure identification, author attribution, or citation graph analysis) ignoring altogether the semantic import of the content expressed using the symbolic language. In scenarios involving proof interpretation, in which constructing a semantic representation of content is *the* computational task, bringing the two languages together is a sine qua non. Yet, as we had previously pointed out, existing systems for processing mathematical discourse do not analyse the symbolic content at all (see (Jeschke et al., 2008) and the overview in Section 1.3.3, p. 47) or merely gloss over phenomena related to the interaction of natural language and mathematical notation (see (Zinn, 2004)).

In this work, we propose a method of achieving a systematic analysis of the mixed language by viewing the symbolic expressions within utterances at the level of their *syntactic types* and treating these types on a par with natural language. To achieve this, processing symbolic mathematical expressions embedded within utterances comprises three subtasks:

- **Identification**, that is, delimiting symbolic expressions within the natural language text,
- **Parsing and annotation**: analysing their structure and semantics and marking the relevant information on the expressions' derivation trees,
- **Interpretation in context**, that is, integrating the symbolic expressions into the syntax and semantics of the utterances in which they appear.

The identification subtask is clear: the purpose of this process is to recognise mathematical expressions within the surrounding natural language text. As we pointed out when discussing parameters in Section 5.2.1, how this process is performed depends on the language of the input contributions, on the mathematical subarea of the discourse, and on the encoding format of the mathematical symbols. Once identified, every mathematical expression is parsed by a mathematical expression parser. In the approach we propose, the parser performs four tasks: it constructs the expression's dependency-style derivation tree,[6] identifies the expression's high-level syntactic type, identifies certain salient substructures, and annotates the derivation tree with the type and substructure information. We distinguish eight types of mathematical expressions. The two obvious basic types are TERM and FORMULA. Their definitions are standard: TERM is the type of ontological mathematical objects.

[6] See Figure 3.2 (p. 98) and the discussion in Section 3.2.1.2 (p. 95).

FORMULAS are sentences, expressions with a truth value. The remaining six types are derived from the basic two and account for incomplete expressions. We will return to those in Section 6.1.3. Once an expression's derivation tree has been constructed, its root node is annotated with information about the expression's type. We also annotate nodes which head visually salient substructures: the head nodes of bracketed subexpressions and the head nodes of the subexpressions to the left and to the right of the root node.[7] This information is relevant for reference resolution which we will discuss in Section 6.3.

Once the mathematical notation is parsed and analysed, parsing utterances with embedded symbolic expressions proceeds based on utterance representations in which the specific mathematical expressions has been abstracted over. As with domain terms, the original mathematical expressions are substituted with tokens which represent their types: mathematical expressions which denote terms are substituted with the token TERM (for instance, $A \cup B$ and $K(A) \cap K(B)$) and those which denote truth values are substituted with the token FORMULA (for instance, $A \cup B = B \cup A$); likewise, partial expressions are substituted with their respective tokens. These tokens are, in turn, represented in the parser's lexicon. In the course of syntactic and semantic parsing, the parser operates on the pseudo-lexemes, and not on the original mathematical expressions; more details follow in Section 5.2.3.1 of this chapter and in Sections 6.1.2 and 6.1.3 of the next chapter. This approach is superior to the one proposed by Zinn of encoding every lexeme of the mathematical vocabulary as part of the utterance parser's lexicon: in our approach the two parsing tasks, which can be performed independently, are clearly separated, thereby improving modularity of the overall architecture and reducing the complexity of the utterance parsing grammar.

The mathematical expression parser implemented for the purpose of the evaluation in Chapter 7 takes word-tokenised text as input and finds mathematical expression substrings using regular expressions. Identification of mathematical expressions within natural language text is based on: single character tokens (including parentheses), multi-character tokens consisting only of known relevant characters, mathematical symbol codes (unicodes and LaTeX-commands in C-I and C-II, respectively), and newline characters. Multi-character candidate tokens are further segmented into operators and identifiers by inserting the missing spaces. A basic precedence-based parser which builds dependency-style tree representations of the mathematical expressions found in the corpora has been implemented. The parser uses

[7] Recall the discussion on the structure of mathematical expressions in Section 3.2.1.2 (p. 95).

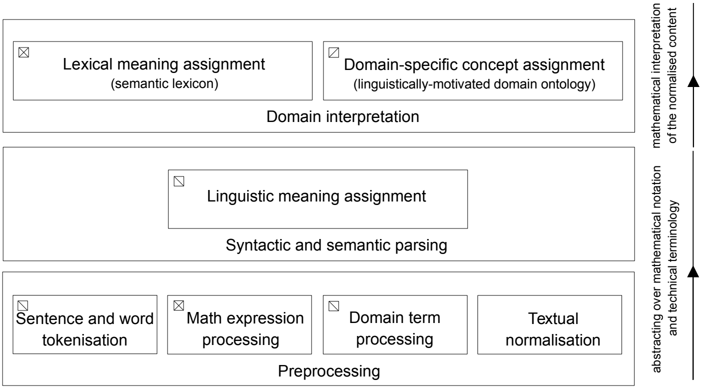

Lexical meaning assignment
(semantic lexicon)
Domain-specific concept assignment
(linguistically-motivated domain ontology)
Domain interpretation
Linguistic meaning assignment
Syntactic and semantic parsing
Sentence and word tokenisation
Math expression processing
Domain term processing
Textual normalisation
Preprocessing
mathematical interpretation of the normalised content
abstracting over mathematical notation and technical terminology

Figure 5.3: Interpretation strategy for informal mathematical language

a knowledge resource with information about all the mathematical symbols used by the learners in both corpora. We also implemented a correction procedure for ill-formed expressions, based on typical errors found in mathematical expressions constructed by students (see Section 3.2.1.5, p. 106). A preliminary evaluation of the algorithm will be presented in Section 6.4 of the next chapter. As with domain terms, for the purpose of the analyses and evaluation presented in Chapters 4 and 7 the formula parser's outputs were verified and corrected by hand. In principle, an external component could be integrated into the implemented processing architecture, so long as for every mathematical expression it can provide its type (FORMULA, TERM or the fragment expression types) as well as access functions to retrieve meaningful subcomponents of symbolic expressions (left-/right-hand side, (nested) bracketed subexpressions, etc.)

5.2.3 Core interpretation strategy for proof discourse

Basic processes involved in understanding informal proof language are (i) syntactic and semantic parsing of proof contributions viewed as linguistic discourses, independently of their specialised domain, whose goal is to construct representations of the contributions' linguistic meaning, and (ii) interpretation of the linguistic meaning representations within the domain (domain of proving in general on the one hand and, on the other hand, the specific mathematical domain with which the given proof is concerned) and in the context of prior discourse. Once a domain interpretation is found, the interpreted semantic representations can be translated into formal representations which serve as input to a domain reasoner.

The complete utterance-level interpretation process is represented schematically in Figure 5.3. In the following sections we present the two core processing steps and a walk-through analysis of a typical utterance from the first corpus (C-I). We focus here on a general strategy for processing the sublanguage of informal mathematical discourse in which natural language and symbolic expressions can be interleaved.

5.2.3.1 Parsing

The first stage of interpretation consists of syntactic and semantic analysis of the proof contributions. The task of the syntactic–semantic parser is to construct representations of the *linguistic meaning* of utterances and syntactically well-formed language fragments. As the linguistic meaning we understand an encoding of the content of an utterance which represents the utterance's decontextualised semantics, where by ''decontextualised'' we mean meaning independent of the domain of discourse, the context in which the utterance appears, of the utterance's intentional content, and illocutionary force. In this sense, linguistic meaning can be thought of as the literal reading of an utterance perceived without reference to any special knowledge of the situation in which the utterance was observed.

Linguistic meaning representation To represent the linguistic meaning we adopt the notion of tectogrammatics, the Functional Generative Description's (FGD) representation of the utterance's semantic dependency structure. FGD is a linguistic theory and a formal grammar formalism being developed by the Prague School of linguistics since the 1960s (Sgall et al., 1986). At the heart of the framework is the notion of *dependency*, originally due to Tesnière, which describes subordination relations between the words in an utterance. Building on Tesnière (1959)'s work, FGD views the utterance in terms of interlinked layers of description which correspond to different levels of meaning: morphological, analytical (surface syntax), and tectogrammatical (deep syntax/semantics). The tectogrammatical layer is conceptually related to logical form, however, differs in coverage: while it does operate at the level of deep semantic roles and accounts for topic–focus articulation, it does not address such aspects of meaning as, for instance, the interpretation of plurals and does not resolve the scope of quantifiers or negation.

In FGD the central unit of description is a *valency frame*, a structure which consists of an autosemantic lexical unit (a verb, a noun, or an adjective, for instance) which constitutes the frame's head, and a set of its possible obligatory and optional complementations, that is, syntactically dependent autosemantic units in certain relations to the head. The head of a valency frame explicitly specifies the *tectogrammatical relations* (TR) of its dependents (or ‘‘participants’’, in the Praguian terminology). A distinction is drawn between *inner participants* and free (adverbial) *modifications*, also called ‘‘functors’’. Inner participants of a valency frame (arguments; corresponding to theta roles, deep cases, or Tesnière’s actants), are the lexeme-specific arguments of the head. Five types of inner participants are distinguished (Sgall et al., 1986):[8]

Actor The ‘‘first actant’’, the agent performing an action or the bearer of a property (‘‘a cat sleeps’’),

Patient/ Objective The object affected by the action and the primary function of the direct complement of a verb, (‘‘to pet a cat’’),

Addressee The primary function of the indirect object (‘‘to give a cat to a child),

Origin The source or initial state of an object (‘‘to let a cat out of a bag’’),

Effect The effect of an action; a primary function of a predicative complement of verbs such as ‘‘nominate’’, ‘‘elect’’, or a result adverbial (‘‘to choose a cat as a pet’’).

Free modifications (adjuncts or circumstantials) express additional information about the head. A large set of free modifications has been proposed for English (Hajičová et al., 2000; Hajičová, 2002). The most common include:

- Locative and directional modifications, such as *Location*, *Where to*, *Where from*;
- Modifications expressing manner: *Extent*, *Means*, *Regard*, *Norm* (‘‘to act in accordance with the law’’, ‘‘to build a machine after a model’’), *Criterion* (‘‘according to the weather report . . . ’’);
- Causal modifications: *Cause* (‘‘. . . because . . . ’’), *Condition* (‘‘If . . . , then . . . ’’), *Aim*, *Result*, *Concession*; these relations may be also realised by prepositional phrases, for instance, ‘‘for personal reasons’’ (Cause), ‘‘under the circumstances’’, ‘‘in this case’’ (Condition), ‘‘for the sake of clarity’’ (Aim);
- Temporal modifications: *When*, *Since when*, *Till when*, *How long*, *For how long*;
- Rhematisers and sentence adverbials: *Modality*, *Attitude*;
- Paratactic construction functors: *Apposition*, *Conjuction*, *Disjunction*.

[8] In the examples, the fragment which contains the dependent node in the given relation to the head is underlined.

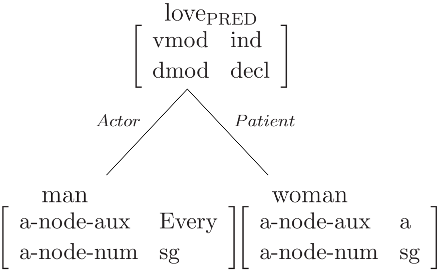


Figure 5.4: Simplified tectogrammatical tree of the sentence ‘‘Every man loves a woman’’

Valency and modification concerns not only verbs, but also nouns, adjectives, and some adverbs. Among free modifiers occurring with nouns there are, for instance, *Identity* (‘‘the notion of identity’’, ‘‘the steamboat Titanic’’), *Material* (‘‘a cup of coffee’’), or *Appurtenance* (‘‘the dog of my cat’s’’). Participants and free modifications can be obligatory or optional. Inner participants are prototypically obligatory and only one inner participant of a given type is allowed to cooccur with one head. Free modifications are prototypically optional. A tectogrammatical dependency structure is a tree with the semanteme which represents the head of an utterance at the root, and with dependent arguments' semantemes at the linked nodes. Only autosemantic words (content bearing words) are represented as nodes of the tectogrammatical layer. Function words are typically represented as attributes of the relevant content words. The nodes (or edges) are labelled with the tectogrammatical relations in which they stand to their directly superordinate nodes.

Figure 5.4 shows an example of a simplified tectogrammatical analysis of the notorious linguistic example: ‘‘Every man loves a woman’’. The lemma ‘‘love’’ is the main predicate (PRED) and the root of the tectogrammatical layer. The valency frame of the transitive verb ‘‘love’’ specifies two participants: an *Actor*, here filled by the lexeme ‘‘man’’ and a *Patient*, here filled by ‘‘woman’’. The node contains grammateme information on the verb's mood (vmod: indicative) and deontic modality (dmod: declarative). The nodes representing both dependents contain references to the analytical layer's auxiliary nodes' information about the quantifier and indefinite modification (a-node-aux), as well as to morphological information about the number (a-node-num).[9]

[9] For a formal definition of tectogrammatics, see (Sgall et al., 1986, p. 150). The tree description presented here is somewhat simplified. For instance, in treebank annotation, a technical node for the tree's root is introduced, which we omitted here. In annotated corpora, references to

The tectogrammatical relations which we use in the semantic representations, unlike surface grammatical roles, provide a generalised view of the relation between (domain-specific) content and the linguistic realisation. To derive our set of semantic relations we generalised and simplified the collection of Praguian tectogrammatical relations in (Sgall et al., 1986; Hajičová et al., 2000). The main reason for this simplification is that certain relations need to be understood metaphorically in the mathematical domain.

The most commonly occurring relations in our domain are *Cause*, *Condition*, and *Result-Conclusion* which coincide with rhetorical relations in the argumentative structure of the proof:

(61) Da [$A \subseteq K(B)$ gilt]$_{\mathit{Cause}}$ alle x, die in A sind sind nicht in B
Because $A \subseteq K(B)$ holds all x which are in A are not in B

(62) Wenn [$A \subseteq K(B)$]$_{\mathit{Condition}}$ dann $A \cap B = \emptyset$
If $A \subseteq K(B)$ then $A \cap B = \emptyset$

(63) Somit ist [. . .]$_{\mathit{Result}}$
With this it holds that . . .

Justifications of inference we interpret as *Criterion* relations:

(64) [nach deMorgan-Regel-2]$_{\mathit{Criterion}}$ ist $K((A \cup B) \cap ...)=...)$
according to De Morgan rule 2 it holds that ...

(65) $K((A \cup B))$ ist [laut DeMorgan-1]$_{\mathit{Criterion}}$ $(K(A) \cap K(B))$
. . . equals, according to De Morgan rule1, . . .

Other relations are grouped into the classes *HasProperty* and *GeneralRelation* (for adjectival and clausal modification), for example:

(66) dann muessen alla A und B [in C]$_{\mathit{HasProperty\text{-}Location}}$ enthalten sein
then all A and B have to be contained in C

(67) Alle x, [die in B sind]$_{\mathit{GeneralRelation}}$ . . .
All x that are in B . . .

(68) alle elemente [aus A]$_{\mathit{HasProperty\text{-}From}}$ sind in $K(B)$ enthalten
all elements from A are contained in $K(B)$

where *HasProperty-Location* denotes a *HasProperty* relation of type *Location*, *GeneralRelation* is a general relation, as in relative clause complementation,

the analytical layer's annotations are used instead of the actual forms. In general, because FGD analysis as such is not our focus, here and in further examples we simplify the representations and omit a lot of information which constitutes part of FGD analyses. We do not show the analytical layer and the links to the tectogrammatical layer. At the tectogrammatical layer we omit morphological grammatemes as well as information on topic–focus articulation. Detailed guidelines on tectogrammatical annotation for English can be found in (Cinková et al., 2006).

Table 5.1: Example categories of Categorial Grammar

Linguistic category	CG category
Sentence	S
Noun phrase	NP
Intransitive verb	S\NP
Transitive verb	(S\NP)/NP
Ditransitive verb	((S\NP)/NP)/NP
Adjunct	S/S

and *HasProperty-From* is a *HasProperty* relation of type *Direction-From* or *From-Source*. All relations which do not need to be translated into a formal representation are grouped in the category *Other*.

Meaning construction with Combinatory Categorial Grammar To construct the linguistic meaning representations we use Combinatory Categorial Grammar; more precisely, Multi-Modal Combinatory Categorial Grammar. We built a lexically specified grammar for a fragment of German and use an open source CCG parser to directly construct semantic dependency representations analogous to those of the tectogrammatical level described above.

Categorial Grammars are a family of syntactic theories and grammar formalisms which are closely related to Dependency Grammars in that both stem from research on type theory and category theory. Foundation which lead to the development of CGs was laid by Leśniewski, Ajdukiewicz, Husserl, and Russell in the 1920s and 30s, and was extended by Bar-Hillel and Lambek in the 50s. CGs explicitly define syntax in the lexicon by associating lexical units of a language with categories of two types: elementary (atomic) types and complex (functional) types which are built up using a category-building operator (denoted with a slash). When modelling linguistic data the types might encode syntactic information on predicate–argument structure, subcategorisation, word order of the object language, etc. Table 5.1 shows examples of atomic categories associated with sentences and nouns and functional categories of English verbs and adjuncts (sentential modifiers).[10]

In the Type Logical, or deductive, tradition of Categorial Grammar, which builds on the Lambek calculus and van Benthem's and Moortgat's categorial

[10] We use the so-called result-first notation for syntactic categories. The signs $\alpha\backslash\beta$ and α/β denote functional types from β to α, where the location of the argument, β, is indicated by the direction of the slash: left (\) or right (/) of the functor α, respectively. The sign $\alpha\backslash\beta$ is thus to be interpreted as forming a category α if an argument of category β is found immediately to its left.

systems (Lambek, 1958; van Benthem, 1987; Moortgat, 1988), parsing is viewed as deduction. On this view, the slash, which builds up partial categories, is considered as a kind of a logical implication operator. The slash (and other operators) together with a set of axioms (inference rules) define a proof theory. For instance, the application rule (slash elimination) corresponds to the Modus Ponens rule of classical logic. Examples of basic inference rules of type logical grammar are shown in Table 5.2. Parsing, that is, determining whether a linguistic expression is well-formed, amounts to finding a proof in the proof system of the given categorial logic.

Combinatory Categorial Grammars, due to Szabolcsi and Steedman, are based on a set of explicitly specified combinatory rules, called *combinators*, which govern the deviation of syntactic structures (Szabolcsi, 1992; Steedman, 2000). The basic set of combinators includes forward and backward directional variants of the rules of functional application, composition, and type-raising; the forward and backward directions are applicable to an argument to the right and left of a functor, respectively. Their schemata are presented in Table 5.3.[11] Multi-Modal Combinatory Categorial Grammar (MMCCG) refines the CCG framework by introducing a means of controlling application of combinatory rules (Baldridge, 2002). Control of rule application is achieved by specifying ''modes'' on category forming operators, the slashes, and making application of rules dependent on the slash mode. There are four hierarchically organised basic modes which govern different levels of associativity and permutativity between signs. The mode $*$ is the most restrictive, allowing only functional application between adjacent signs. The modes $\diamond$ and $\times$ allow associative, non-permutative (harmonic) and permutative, non-associative (crossed) composition, respectively. The mode $\bullet$ is the least restrictive and allows application of all combinatory rules.[12] Figure 5.5 shows an example derivation of the sentence ''Every man loves a woman'' in CCG. Figure 5.6 illustrates blocking the derivation of an ungrammatical fragment ''a good from Bordeaux wine'' (from (Baldridge and Kruijff, 2003)) in MMCCG. The mode $*$, more restrictive than $\diamond$, prevents modifiers in invalid order from being combined.[13]

We argue that CCG, or CG in general, is an appropriate framework for modelling syntactic language phenomena in mathematical discourse. The motivation for this approach is two-fold: First, categorial grammar

[11] There is a strong analogy between the inference rules of the type logical categorial grammar system and the combinators of combinatory categorial grammars; see (Steedman, 2000) for details.

[12] In the following examples of syntactic categories we consider the $*$ mode as default, that is, unless a slash is marked with a specific mode, the functional application mode is assumed.

[13] The grammars have been implemented in OpenCCG. (`http://www.opennlp.org`)

Table 5.2: Basic deduction rule schemes of Type Logical Categorial Grammar

Rule	Schemes	
Lexical instantiation	$\dfrac{e}{A}\ \mathrm{Lx}$	
Slash elimination	$\dfrac{\overset{\vdots}{A/B} \quad B}{A}\ /\mathrm{E}$	$\dfrac{\overset{\vdots}{B} \quad A\backslash B}{A}\ \backslash\mathrm{E}$
Slash introduction	$\dfrac{\dfrac{\vdots \qquad [A]^n}{B}}{B/A}\ /\mathrm{I}^n$	$\dfrac{\dfrac{[A]^n \qquad \vdots}{B}}{B\backslash A}\ \backslash\mathrm{I}^n$

Table 5.3: Basic combinatory rules of Combinatory Categorial Grammar

Rule	Schemes			
Application	(>)	$X/Y \;\; Y \Rightarrow X$	(<)	$Y \;\; Y\backslash X \Rightarrow Y$
Composition	(>**B**)	$X/Y \;\; Y/Z \Rightarrow X/Z$	(<**B**)	$X\backslash Y \;\; Z\backslash X \Rightarrow Z\backslash Y$
Type-raising	(>**T**)	$X \Rightarrow Y/(Y\backslash X)$	(<**T**)	$X \Rightarrow Y\backslash(Y/X)$

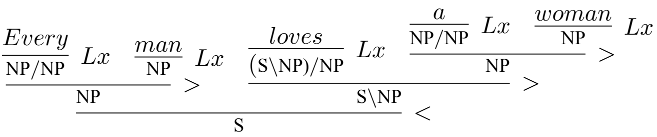


Figure 5.5: Combinatory Categorial Grammar derivation of the sentence ‘‘Every man loves a woman’’

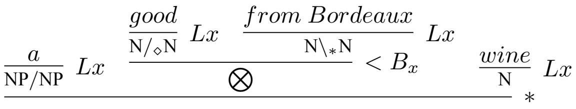


Figure 5.6: Blocking ungrammatical derivation using modes on slashes in MMCCG

is a recognised formalism which enables modelling complex linguistic phenomena. It is known for its account of coordination phenomena (Steedman, 2000), widely present in mathematical discourse, and word order phenomena; see, for instance, (Hepple, 1990; Steedman, 2000; Baldridge, 2002). Moreover, CCG accounts of various word order phenomena in Germanic languages have been proposed; see, for instance, (Carpenter, 1998; Steedman, 2000; Hockenmaier, 2006; McConville, 2007). Second, and most importantly in the case of mathematical discourse, mathematical expressions, represented as their types, lend themselves to a perspicuous categorial treatment described below.

An approach to interleaved symbolic and natural language As mentioned earlier, in the course of parsing, we treat symbolic tokens, which represent types of mathematical expressions (see Section 5.2.2.3), on a par with natural language lexical units. Within utterances, mathematical terms typically occur in the syntactic functions of nouns or noun phrase categories, while mathematical formulas are syntactically sentences or clauses. In the parser's lexicon we encode ''generic'' lexical entries (pseudo-lexemes) for each mathematical expression type together with information on the plausible syntactic categories which expressions of the given type may take. The basic mathematical lexemes in our grammar are TERM and FORMULA. For mathematical expressions denoting terms, represented as TERM lexemes, we encode the noun and noun phrase categories, N and NP, while for truth-valued expressions, FORMULA lexemes, we encode the category of a sentence, S, as the following two examples illustrate:

TERM	equals	TERM
NP	(S\NP)/NP	NP

If	FORMULA	then	FORMULA
(S/S)/S	S	(S\(S/S))/S	S

A number of further atomic and partial categories are defined in the grammar for mathematical expression types in order to account for more complex interactions between mathematical notation and the linguistic material within which it can be embedded. We will return to these in Section 6.1 (p. 206). The choice of syntactic categories associated with mathematical expression tokens was guided by analysing syntactic contexts in which mathematical expressions are used in our corpora and in mathematical textbooks and publications.

The semantic language Aside from syntactic analysis, the parsing framework we use to analyse proof language builds semantic representations of the input utterances. The semantic forms reflect the tectogrammatical structure of the utterances and are encoded using a formal language capable of capturing the relational nature of the tectogrammatical dependency representations.

The linguistic meaning, built in parallel with the syntactic derivation, is represented using Hybrid Logic Dependency Semantics (Baldridge and Kruijff, 2002, 2003). HLDS is a fragment of the language of hybrid logic (Blackburn, 2000) developed specifically to represent natural language semantics in terms of dependency relations. In this work we do not use HLDS as logics; we use it merely as a representation language for the relational structures of dependency-based semantics. Dependency relations of tectogrammatical structures are encoded as modal relations, denoted with the modal logic operator $\langle\rangle$. Each dependent is associated with a nominal, d, which also represents its discourse referent. Predicates, tectogrammatical PREDs, correspond to propositions and form the head, h, of HLDS terms. The notation is illustrated below (after (Baldridge and Kruijff, 2002)):

$$@_h(\textbf{proposition} \wedge \langle\delta_i\rangle(d_i \wedge \textbf{dep}_i))$$

δ ranges over the set of tectogrammatical relations, a referent d_i is created for each autosemantic lexeme, $\textbf{dep}_i$, at the tectogrammatical level. Given this notation, the linguistics meaning of the sentence ‘‘Ed read a red book in London’’ is represented as:

$$\begin{aligned} @_h(\textbf{read} \wedge \langle \textit{Actor}\rangle(d_0 \wedge \textbf{ed}) \\ \wedge \langle \textit{Patient}\rangle(d_4 \wedge \textbf{book} \wedge \langle \textit{GeneralRelation}\rangle(d_3 \wedge \textbf{red})) \\ \wedge \langle \textit{Location}\rangle(d_6 \wedge \textbf{london})) \end{aligned}$$

As explained earlier, the linguistic meaning of an utterance is context- and domain-neutral: it represents the literal interpretation of the utterance semantics. That is, the semantic representations built at the parsing stage do not contain any information as to how the utterance is to be interpreted in the context of the given domain. In order to place the meaning representations in the context of the proving task and the domain of mathematics, the elements of the semantic representations, the terms and relations of the logical forms, are further interpreted using lexical and domain-specific resources.

5.2.3.2 Domain interpretation

The interpretation process in our approach gradually enriches (‘‘annotates’’) the linguistic meaning representations with information stemming from domain resources. Interpretation is a stepwise procedure in which predicates and

Table 5.4: Example entries from the semantic lexicon

TR structure		Lexical meaning
(**equal**$_{\text{PRED}}$, *Actor*$_x$, *Patient*$_y$)	:=	(*Equality*, *Object*$_x$, *Object*$_y$)
(**hold**$_{\text{PRED}}$, *Actor*$_p$)	:=	(*Claim*, p)
(FORMULA$_{\text{PRED},p}$)	:=	(*Claim*, p)
(*Criterion*$_x$)	:=	(*Evidence*, x)
(**p1**$_{\text{PRED}}$, *Reason*$_{p2}$)	:=	(*Reason*, p1, p2)
(**p1**$_{\text{PRED}}$, *Condition*$_{p2}$)	:=	(*Condition*, p1, p2)

relations of the tectogrammatical dependency representations are assigned domain- and task-specific semantics. Task-specific interpretation concerns the meaning in the context of the task of theorem proving, while by domain-specific semantics we mean semantics in the context of the mathematical domain(s) with which the given proof is concerned; set theory or binary relations in the case of our two corpora.

First, semantemes and relations of the tectogrammatical frames are mapped to concepts through a language-specific *semantic lexicon*. The mapping serves either to assign the elements of tectogrammatical frames predicates and roles which denote domain concepts, or provides procedural ‘‘meaning recipes’’ for computing lexical meanings. This is done by associating dependency frames output by the parser with linguistically-motivated domain-relevant conceptual frames represented in a semantic lexicon. The input structures of the semantic lexicon are described in terms of tectogrammatical valency frames of lexical items which evoke given concept(s) or in terms of information on which elements of dependency structures need to be retrieved in order to recover the lexical meaning. The output structures are either the evoked concepts with roles indexed by tectogrammatical frame elements or results of executing ‘‘interpretation scripts’’, operations on dependency structures which enable to recover the lexical meaning. Where relevant, sortal information for role fillers is also given. Example basic entries from lexicon are shown in Table 5.4. Consider the fourth and fifth entries: the *Criterion* tectogrammatical relation introduces the concept of *Evidence*, with the dependent in the *Criterion* relation expressing the actual evidence according to which the head proposition holds, the *Reason* tectogrammatical relation is interpreted as expressing a *Reason* for an eventuality, with the daughter dependent actually specifying the reason. An example of a procedural recipe is the representation of the adjective ‘‘gemeinsam’’ (*common*) or of the semantically complex adverb ‘‘umgekehrt’’ (*the other way (a)round*) which will be shown in Chapter 6.

Next, the concepts are interpreted within the mathematical domain using a manually constructed intermediate domain model. The model is a *linguistically-motivated domain ontology*, a hierarchically organised representation of domain objects and relations along with their properties, which enables limited reasoning about relations between objects; for instance, type checking. It provides a link between the conceptual frames evoked by lexical items encoded in the semantic lexicon and domain-specific (here: mathematical) concepts. For instance, the concept of *Evidence* is linked via the relations ontology to the relation `Justification` in the mathematical domain of proofs. The purpose of the ontology as an intermediate representation is also to mediate between the discrepant views of linguistic analysis and deduction systems' representation (see also the discussion in (Horacek et al., 2004)). The domain-specific objects from the ontology could be, in principle, further linked to their logical definitions in a mathematical knowledge base, such as MBase (Kohlhase and Franke, 2001).[14] The motivation for using an intermediate representation instead of directly accessing a mathematical knowledge base will become clear when we discuss imprecision and ambiguity in Section 6.2. More details on the domain model and examples of the modelled objects and relations will be also presented in Chapter 6.

To summarise, as a result of the interpretation process, semantic dependency structures of input contributions are ‘‘annotated’’ with gradually more specific semantic information first at the level of domain-independent concepts, and then (possibly ambiguous) domain-specific interpretations. Two points need to be kept in mind: First, if multiple readings are found, the language interpretation module alone is *not* in a position to identify the one that is plausible in the given proof context. In particular, linguistic meaning ambiguity may lead to both logically correct and incorrect proof steps. (Consider, for instance, the utterance ‘‘FORMULA if and only if FORMULA and FORMULA’’.) All parses are assigned an interpretation by the language understanding component and are passed on to a reasoner. It is also plausible to assume that disambiguation could be performed at the dialogue level, before evaluation, by asking an explicit clarification question. In the case of a structurally ambiguous pattern such as ‘‘FORMULA if and only if FORMULA and FORMULA’’, the system could ask, for instance, ‘‘Do you mean ‘... if and only if ... *and moreover* ... holds’ or ‘... if and only if *both* ... *and* ... hold’?’’ In the dialogue in which the utterance ‘‘... genau dann wenn ... und ...’’ appeared, the tutor did not clarify the intended reading and accepted the proof step, that is, cooperatively assumed that the

[14] This link has not been realised as part of this thesis.

correct interpretation was intended. (Or, possibly, did not even realise that ambiguity was present.) For a tutoring system, one option would be to take the same strategy: if at least one reading yields a correct step, this reading could be assumed to be intended. Another option would be to leave the decision whether to accept an ambiguous step to the pedagogical module which could, in turn, refer to its student model to decide on appropriate action. Modelling this decision is outside of the scope of this thesis.

Second, within the annotated HLDS terms only the *linguistically realised* content is represented and the language processing system is not in a position to reason about its validity nor to fill in omitted proof step components. However, the annotated dependency structures can be transformed (rewritten) into representations for further processing, for instance, by an automated theorem prover. In the tutoring system's architecture presented in Section 1.2 this is the task of the Proof representation processing module (see p. 39).

5.3 A walk-through example

As an illustration of the interpretation process, we give a step by step analysis of utterance (6), reproduced below, which is a typical utterance from C-I:

(69) $K(A \cup B)$ ist laut DeMorgan-1 $K(A) \cap K(B)$
$K(A \cup B)$ is according to DeMorgan-1 $K(A) \cap K(B)$

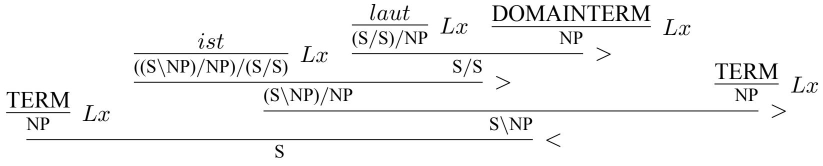
As a result of preprocessing, the utterance is transformed into a form that abstracts away from the mathematical expressions and concrete domain terms:

TERM ist laut DOMAINTERM TERM

The categories, encoded in the grammar, which correspond to the words in the utterance are:

TERM	:=	NP
ist	:=	((S\NP)/NP)/(S/S)
laut	:=	(S/S)/NP
DOMAINTERM	:=	NP

The abstracted form is parsed using the CCG parser as follows:



The linguistic meaning representation constructed by the parser consists of the German copula, ‘‘ist’’, with the symbolic meaning **equal** as the head of the dependency structure, and three dependents in the tectogrammatical relations *Actor*, *Criterion*, and *Patient*. The HLDS term corresponding to this dependency structure is shown below:

$$\begin{aligned}@_i(\mathbf{equal} \wedge \langle \mathit{Actor} \rangle (d_1 \wedge \mathbf{TERM})\\ \wedge \langle \mathit{Patient} \rangle (d_5 \wedge \mathbf{TERM})\\ \wedge \langle \mathit{Criterion} \rangle (d_4 \wedge \mathbf{DOMAINTERM}))\end{aligned}$$

Stepwise domain meaning assignment proceeds as follows: First, based on the semantic lexicon, a concept *Equality* is assigned to **equal**, with the *Actor* and *Patient* dependents as relata, and the *Criterion* dependent is interpreted as an *Evidence*. Next, *Equality*, in the context of set theory TERMs, is interpreted as `Set equality`, and *Evidence*, in the context of theorem proving, as a `Justification` in a proof step. A simplified presentation of the entire interpretation process is shown schematically in Figure 5.7 (p. 202).

5.4 Summary

This chapter outlined an architecture for processing informal mathematical proof discourse such as that found in tutorial dialogues. The design of the architecture was motivated by the goal of processing not only students’ input in tutorial dialogues, but also narrative discourse such as that found in textbooks or mathematical publications. This goal has been achieved by modularisation of the system’s components while taking into account the peculiarities of mathematical language: its two ‘‘modes’’ (natural language interleaving with mathematical notation) and the presence of technical vocabulary (single and multi-word domain terms). While mathematical notation itself is analysed by a dedicated module and not by the natural language parser, the information identified by the mathematical expression parser is used to encapsulate the specific instances of notation in terms of pseudo-lexemes, denoting the expressions’ types, which are encoded in the natural language parser’s lexicon. Likewise, specialised terminology is recognised by a dedicated module and domain term instances are encapsulated in pseudo-lexemes. Modularisation of this kind facilitates efficient management of system resources: depending on the mathematical subarea of discourse, an appropriate mathematical expression parser or domain lexicon can be integrated without changes to the overall system. By abstracting over the symbolic notation and domain terminology we moreover ensure that the adaptation of the natural language parser when switching to

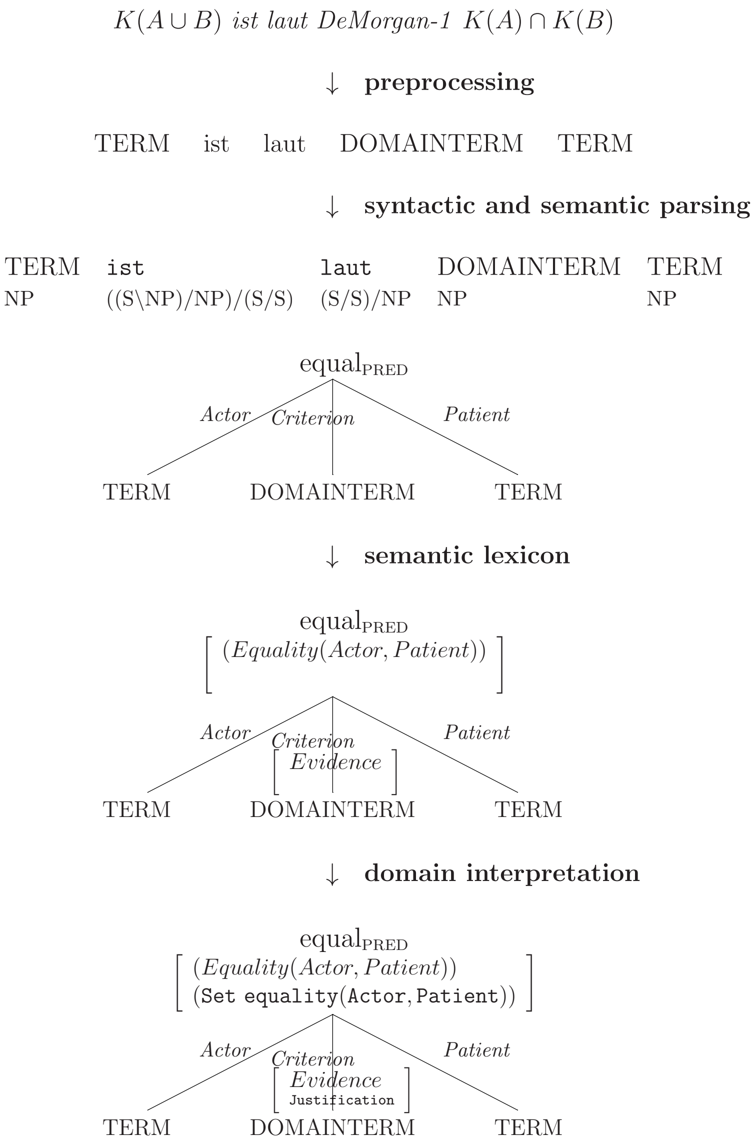

$K(A \cup B)$ *ist laut DeMorgan-1* $K(A) \cap K(B)$
↓ **preprocessing**
TERM ist laut DOMAINTERM TERM
↓ **syntactic and semantic parsing**
TERM NP
ist ((S\NP)/NP)/(S/S)
laut (S/S)/NP
DOMAINTERM NP
TERM NP
$\mathrm{equal}_{\mathrm{PRED}}$
Actor
Criterion
Patient
TERM
DOMAINTERM
TERM
↓ **semantic lexicon**
$\mathrm{equal}_{\mathrm{PRED}}$
$(Equality(Actor, Patient))$
Actor
Criterion
Patient
Evidence
TERM
DOMAINTERM
TERM
↓ **domain interpretation**
$\mathrm{equal}_{\mathrm{PRED}}$
$(Equality(Actor, Patient))$
(Set equality(Actor, Patient))
Actor
Criterion
Patient
Evidence
Justification
TERM
DOMAINTERM
TERM

Figure 5.7: Interpretation process for the utterance "$K(A \cup B)$ ist laut DeMorgan-1 $K(A) \cap K(B)$" (notation, semantic lexicon, and ontology entries simplified)

a new mathematical domain is limited as much as possible to extending the parser's coverage of syntactic constructions, rather than its vocabulary, thus minimising out-of-vocabulary parser errors. As we will show in Chapter 7 this approach and the choice of categorial grammar over a simpler formalism results in good scalability of the parsing process.

The basic processing strategy presented in this chapter covers the most prominent language phenomena found in mathematical utterances: (i) the most common syntactic categories of mathematical expressions embedded within natural language utterances: terms as nouns or noun phrases and formulas as sentences/clauses, (ii) the basic syntax of mathematical language found in our corpora as well as in typical textbook proofs (for instance, constructions such as ''Wenn FORMULA dann FORMULA'' (*If FORMULA, then FORMULA*) or ''Deshalb FORMULA'' (*Therefore, FORMULA*)), (iii) the basic syntactic categories of the most frequent verbal constructions (such as ''gelten'' (*hold*) or ''(gleich) sein'' (*be equal (to)*), etc.), and (iv) the semantics of constructions which can be directly interpreted in the context of proofs and within the domains of naïve set theory and binary relations (for instance, the *Criterion* or *Reason* relations, which need to be interpreted as a justification of a proof step, or the meaning of basic verbal constructions, such as those mentioned above). However, the mixed, natural and formal-symbolic, language and the informality of the mathematical discourse in our setting require extensions to the basic analysis strategy in order to account for a wider range of linguistic phenomena and, in particular, to enable *cooperative* interpretation. By ''cooperative'' we mean that, for instance, certain non-canonical syntactic structures or domain-specific readings of common words should be interpreted without resorting to signalling non-understanding, requesting repair, or entering a clarification subdialogue. The next chapter presents details on processing a subset of language phenomena found in our corpora and the resources constructed for cooperative interpretation of imprecise language.

Chapter 6

Modelling selected language phenomena in informal proofs

In this chapter we show how selected phenomena identified in the students' contributions can be modelled. As we have shown in Chapters 3 and 4 students' language is complex, rich in linguistic phenomena, and diverse. Modelling all the linguistic phenomena found in our data is out of a scope of one thesis. The selection included in this chapter was motivated by two factors: First, we address those phenomena which systematically recur and are critical for automated proof tutoring, the core scenario and motivation for this thesis, to be feasible. This includes modelling basic syntactic phenomena (German word order in recurring constructions in mathematics, the mixed language, and the syntactic irregularities characteristic of our domain) and basic semantic imprecision phenomena. Second, we also selected a number of interesting phenomena, which are not as highly represented in our corpora, but which did occur, suggesting that they might also reappear in new or other corpora (semantic reconstruction of a certain contextual operator, reference to symbolic notation and propositions, and mathematical expression correction). Because our data is sparse, we designed preliminary algorithms and evaluated them in proof-of-concept evaluations or conducted corpus studies as preliminary step towards algorithm development. The chapter shows that the processing methodology we adopted, in particular, deep parsing using categorial grammars which build domain-independent linguistic meaning representations of the analysed input, lends itself well to modelling a number of phenomena found in students' informal mathematical language. Material presented in this chapter has been published in the following articles: (Wolska et al., 2004a; Wolska and Kruijff-Korbayová, 2004a; Gerstenberger and Wolska, 2005; Horacek and Wolska, 2005a,c; Wolska and Kruijff-Korbayová, 2006b; Horacek and Wolska, 2006a,b,c).

6.1 Syntactic phenomena

The scope of the implemented parser resources, the vocabulary and syntactic categories, are limited to the language in our corpora. Methods of modelling syntactic phenomena – basic German word order, incomplete mathematical expressions used as a form of shorthand for natural language, scope phenomena involving parts of mathematical expressions, and the use of spoken-language syntax to verbalise mathematical expressions – are outlined below.

6.1.1 Basic German word order in Combinatory Categorial Grammar

German is typically described as a ‘‘verb-second’’ language. The placement of the finite verb depends on the clause type (main vs. dependent) and the sentence mood (declarative vs. interrogative vs. imperative). Three types of clauses can be distinguished with respect to the finite verb position: verb-initial, verb-second, and verb-last clauses.

In declarative main clauses, such as (70) below, and wh-questions, (71), the finite verb is in the ‘‘second’’ position. It need not be literally the second word in the sentence, as (70) illustrates, but the second *macrostructural element* (more on this in the section on the Topological Field Model):

(70) Der Mann fuhr den Wagen vor.
The man brought the car round.

(71) Wer fuhr den Wagen vor?
Who brought the car round?

The matrix clause of yes/no questions, (72), and alternative questions as well as imperatives, (73), are verb-first, that is, their finite verb is in the sentence-initial position:[1]

(72) Hat der Mann den Wagen gefahren?
Did the man drive the car?

(73) Fahre den Wagen!
Drive the car!

Other clause types in which finite verbs occur in the first position include verb-initial conditionals, hypotheticals, and formal concessive clauses not introduced by a conjunction (corresponding to the English forms ‘‘Should..., ...’’).

[1] An exception are intonation questions, as in ‘‘Du hast den Wagen gefahren?...’’ (*You drove the car?...*), which may be meant ironically.

Finally, subordinate adverbial clauses, (74), relative clauses, (75), and complementation clauses, (76), exhibit the verb-last pattern:

(74) Wenn Du willst, kannst Du den Wagen fahren.
If you want, you can drive the car.

(75) Maria fährt den Wagen, den der Mann gefahren hat.
Maria is driving the car that the man drove.

(76) Ich glaube, daß Maria den Wagen fahren kann.
I think Mary can drive the car.

Topological Field Model

German clauses are traditionally analysed in terms of *topological fields*, syntactic macrostructures delimited by verbal elements (a finite verb or a verb complex) or clause markers (for instance, a complementiser, a wh- or relative pronoun). The Topological Field Model proposed by Höhle (1983) is a linguistically-motivated theory-neutral description of the macrostructure of the clause, which characterises the clause not from the point of view of phrase structure, but from the point of view of the distributional properties of constituents in the clause with respect to the finite verb. The basic model divides clauses into five macrostructural elements: the Vorfeld (*pre-field*), the Linke Klammer (*left bracket*), the Mittelfeld (*middle field*), the Rechte Klammer (*right bracket*), and the Nachfeld (*post-field*).

Table 6.1 (p. 208) shows the elements of the model and the placement of the different constituent types within the macrostructure.[2] In verb-initial and verb-second clauses, the finite verb occupies the Linke Klammer field. In the verb-final clauses, the finite verb occupies the Rechte Klammer. Not all the fields have to be occupied in a sentence and certain elements are optional. For certain fields there are restrictions on the number and type of constituents which can occur. For instance, German grammar rules restrict the number of constituents in the Vorfeld to at most one. In main declarative clauses this can be an argument of the finite verb, an adjunct, or, in case of complex sentences, a fronted dependent clause. The latter are frequent in mathematical discourse (consider, for instance, ‘‘weil’’-clauses or conditional clauses without the subordinating conjunction). In case of adjuncts of the same semantic type, a cluster of adjuncts is also allowed in the Vorfeld.[3]

[2] From (Wöllstein-Leisten et al., 1997, p. 53).

[3] In certain cases complements of different semantic types may also be fronted together, as in the following sentence from (Müller, 2003): ‘‘Zum zweiten Mal die Weltmeisterschaft errang Clark 1965...’’ (*For the second time Clark became the world champion in 1965...*). The temporal

Table 6.1: Constituent ordering in the Topological Field Model; Optional elements in italics.

Clause type	Vorfeld	Linke Klammer	Mittelfeld	Rechte Klammer	Nachfeld
verb-first		finite verb	*constituents*	*non-finite verb*	*constituents*
verb-second	constituent	finite verb	*constituents*	*non-finite verb*	*constituents*
verb-last		subordinating conjunction	*constituents*	*non-finite* finite verb	*constituents*

Table 6.2: Topological analyses of example German sentences. (Example numbers refer to example numbers in text; ‘‘m’’ denotes a matrix clause, ‘‘s’’ a subordinate clause.)

Example No.	Vorfeld	Linke Klammer	Mittelfeld	Rechte Klammer	Nachfeld
(72)		Kannst	den Wagen	fahren?	
(73)		Fahre	den Wagen!		
(70)	Der Mann	fuhr	den Wagen	vor.	
(71)	Wer	fuhr	den Wagen	vor?	
(74s)		Wenn	du	willst, . . .	
(74m)	Wenn du willst,	kannst	den Wagen	fahren.	
(75m)	Maria	fährt	den Wagen,		den der Mann gefahren hat.
(75s)	. . . den		der Mann	gefahren hat.	
(76m)	Ich	glaube,			daß Maria den Wagen fahren kann.
(76s)		. . . daß	Maria den Wagen	fahren kann.	

In complex sentences, the model is applied to each clause individually: iteratively in paratactically conjoined clauses and recursively in hypotactically conjoined clauses. Table 6.2 (p. 208) shows the topological field analysis of the sentences (72) through (76). For the sentences (74) through (76) both the analysis of the main clauses (marked with ‘‘m’’) and of the subordinate clauses (‘‘s’’) are shown to demonstrate the recursivity of the model in embedded clauses. Examples (77) and (78) below illustrate the word order phenomena based on utterances from the corpora:

(77) [$K(A \cup B)$ ist laut DeMorgan-1 $K(A) \cap K(B)$]$_{\mathrm{V2}}$

(78) [[Wenn alle A in $K(B)$ enthalten sind]$_{\mathrm{VL}}$ und [dies auch umgekehrt gilt]$_{\mathrm{VL}}$,]$_{\mathrm{VL}}$ [muß es sich um zwei identische Mengen handeln]$_{\mathrm{V2}}$

Modelling German word order in CCG

Work on Combinatory Categorial Grammars for Germanic languages often focuses on addressing linguistic phenomena peculiar to this language family, such as cross-serial dependencies in Dutch; see, for instance, (Steedman, 2000). Verb argument fronting has been also discussed, however, for languages like German and Dutch, the phenomenon of fronting concerns not only verb arguments, but also free modifiers (adverbs, adverbial prepositional phrases, etc.) which exhibit the same syntactic behaviour. This phenomenon has been rarely addressed in CCG accounts. Partial free word order in Germanic languages has been modelled by employing language specific combinatory rules. Steedman (2000) and Baldridge (2002) show accounts of verb argument fronting and free modifiers in the sentence-medial position, however, a way of controlling multiple constituents in the sentence-initial position is not shown for free modifiers. The Bielefeld German CCG for human–robot dialogue employs a counting mechanism to check the number of fronted verb arguments as a way for testing which clause type has been derived: if no argument has been fronted then a verb-initial clause has been derived, if there is only one argument fronted then the derived clause is verb-second, etc. (Hildebrandt et al., 1999; Vierhuff et al., 2003). Again, optional adjunct and free modification fronting is not addressed. Carpenter (1998) does account for adverbial fronting by compiling context-specific syntactic categories into the lexicon with appropriate features to control derivation. The approach we present is similar, however, while

adverbial ‘‘zum zweiten Mal’’ (*for the second time*) and a *Goal* dependent of the verb (*reach*), ‘‘die Weltmeisterschaft’’ (*the world championship*), both occur in the Vorfeld here. There are a number of further exceptions to the single Vorfeld constituent rule which account for syntactically marked topic–focus realisation. See, for instance (Müller, 1999; Müller, 2003) for a detailed discussion.

Carpenter populates verb categories by instantiating them for every licensed fronting configuration, our approach attempts to minimise the number of context-specific lexical entries via generalisation exploiting topological field information and a rich set of features marking verb, conjunction, and adjunct categories. In recent work, Vancoppenolle et al. (2011) employ language specific topicalisation rules (type changing rules) which derive verb-second order from verb-first order by fronting a verb argument or an infinitival clause, which allows them to reduce the number of lexical entries even further. Our approach is simpler in that introducing topological field information into the CCG analysis constrains derivations directly in the lexicon. Taking into account clause bracketing formed by the verbal elements (shown in Table 6.2), we model the CCG lexicon in such way that, where relevant, syntactic categories incorporate information about the topological fields of adjacent categories. The following sections outline the basic principles of our lexical category description.

Verb categories In main declarative clauses, the Vorfeld must be non-empty and the number of constituents occupying it is restricted to one. (Recall Footnote 3 on exceptions though). In order to account for these constraints, we mark verb categories, among others, with attributes which indicate the clause type (*cl-type*): main vs. subordinate, and the status of the Vorfeld (VF). The attribute *VF* takes values from the set $\{+, -\}$, where ''$-$'' indicates that there is no material in the VF and ''$+$'' indicates that a verb taking the given category expects material in its left context. Different word order configurations are compiled into the lexicon of the grammar. For example, the syntactic signs of a transitive verb, such as ''fahren'' (*drive*) are the following:[4]

$$
\begin{array}{lll}
\text{fuhr} := & \mathrm{S}\big[\,\mathit{VF}:+,\mathit{cl\text{-}type}:\mathit{main}\,\big]\backslash\mathrm{NP}_{\mathit{Actor}}/\mathrm{NP}_{\mathit{Patient}} & \text{(for SVO word order)}\\
& \mathrm{S}\big[\,\mathit{VF}:+,\mathit{cl\text{-}type}:\mathit{main}\,\big]\backslash\mathrm{NP}_{\mathit{Patient}}/\mathrm{NP}_{\mathit{Actor}} & \text{(OVS)}\\
& \mathrm{S}\big[\,\mathit{VF}:-,\mathit{cl\text{-}type}:\mathit{main}\,\big]/\mathrm{NP}_{\mathit{Actor}}/\mathrm{NP}_{\mathit{Patient}} & \text{(VSO)}\\
& \mathrm{S}\big[\,\mathit{VF}:-,\mathit{cl\text{-}type}:\mathit{main}\,\big]/\mathrm{NP}_{\mathit{Patient}}/\mathrm{NP}_{\mathit{Actor}} & \text{(VOS)}\\
& \mathrm{S}\big[\,\mathit{cl\text{-}type}:\mathit{subord}\,\big]\backslash\mathrm{NP}_{\mathit{Patient}}\backslash\mathrm{NP}_{\mathit{Actor}} & \text{(SOV)}\\
& \mathrm{S}\big[\,\mathit{cl\text{-}type}:\mathit{subord}\,\big]\backslash\mathrm{NP}_{\mathit{Actor}}\backslash\mathrm{NP}_{\mathit{Patient}} & \text{(OSV)}
\end{array}
$$

The first two entries account for fronting verb arguments, the next two allow constituents other than arguments (such as adjuncts, subjunctions, etc.) to occupy the Vorfeld. The last two entries model subordinate clauses. Since subordinate clauses are always verb-last there is no need to control the status of the Vorfeld which in this case is always either empty – see (74s) and (76s)

[4] A number of attributes, such as, person, number, tense, case of the arguments, etc. have been omitted to simplify the presentation.

```
s{cl-type=main, tense=past, num=sg, pers=3rd, vform=fin, VF=+} :
@w2(fahren ^ <Actor>(w1 ^ Mann) ^ <Patient>(w4 ^ Wagen))
------------------------------
(lex) np/^np : @X_0(<det>def)
(lex) np : (@X_6(Mann) ^ @X_6(<num>sg))
(>)   np : (@X_0(Mann) ^ @X_0(<det>def) ^ @X_0(<num>sg))
(lex) s{cl-type=main, tense=past, num=sg, pers=3rd, vform=fin, VF=+}
      \np{case=nom, num=sg, pers=3rd}/^np{case=acc}
      : (@E_12(fahren) ^ @E_12(<Actor>X_12) ^ @E_12(<Patient>Y_12))
(lex) np/^np : @X_18(<det>def)
(lex) np : (@X_24(Wagen) ^ @X_24(<num>sg))
(>)   np : (@X_18(Wagen) ^ @X_18(<det>def) ^ @X_18(<num>sg))
(>)   s{cl-type=main, tense=past, num=sg, pers=3rd, vform=fin, VF=+}
      \np{case=nom, num=sg, pers=3rd}
      : (@E_12(fahren) ^ @E_12(<Actor>X_12) ^ @E_12(<Patient>X_18)
                       ^ @X_18(Wagen))
(<)   s{cl-type=main, tense=past, num=sg, pers=3rd, vform=fin, VF=+}
      : (@E_12(fahren) ^ @E_12(<Actor>X_0) ^ @E_12(<Patient>X_18)
                              ^ @X_0(Mann) ^ @X_18(Wagen))
```

Figure 6.1: Logical form and derivation of the sentence ‘‘Der Mann fuhr den Wagen’’ (OpenCCG output; some parts of derivation omitted for the sake of readability; see p. 197 for the explanation of the semantic notation)

in Table 6.2 – or occupied solely by the relative pronoun – see (75s) in the same table. The derivation of a simple SVO sentence ‘‘Der Mann fuhr den Wagen’’ (*The man drove the car*), shown in Figure 6.1, reflects the attribute marking introduced by the verb entry: the status of the Vorfeld is occupied ($\mathit{VF} : +$) and the clause type is main ($\mathit{cl\text{-}type} : \mathit{main}$). The grammar is also be able to parse the string ‘‘der Mann den Wagen fuhr’’, however the *cl-type* value of the resulting structure will be *subord*, indicating a subordinate clause structure.

Conjunction categories The same mechanism is used to model complex sentences with recursive embedding. Given the marking on verb categories, we model subjunctions such as ‘‘wenn’’ (*if*), ‘‘weil’’ (*because*), see (74s) in Figure 6.2, by setting their syntactic categories as follows:

$$\begin{aligned}
\text{wenn} := {} & \mathrm{S}\left[\,\mathit{VF} : +\,\right]\backslash\mathrm{S}\left[\,\mathit{VF} : +, \mathit{cl\text{-}type} : \mathit{main}\,\right]/\mathrm{S}\left[\,\mathit{cl\text{-}type} : \mathit{subord}\,\right]\\
& \mathrm{S}\left[\,\mathit{VF} : +, \mathit{cl\text{-}type} : \mathit{main}\,\right]/\mathrm{S}\left[\,\mathit{VF} : -, \mathit{cl\text{-}type} : \mathit{main}\,\right]/\mathrm{S}\left[\,\mathit{cl\text{-}type} : \mathit{subord}\,\right]\\
& \mathrm{S}\left[\,\mathit{cl\text{-}type} : \mathit{subord}\,\right]/\mathrm{S}\left[\,\mathit{cl\text{-}type} : \mathit{subord}\,\right]/\mathrm{S}\left[\,\mathit{cl\text{-}type} : \mathit{subord}\,\right]\\
& \mathrm{S}\left[\,\mathit{cl\text{-}type} : \mathit{subord}\,\right]\backslash\mathrm{S}\left[\,\mathit{cl\text{-}type} : \mathit{subord}\,\right]\backslash\mathrm{S}\left[\,\mathit{cl\text{-}type} : \mathit{subord}\,\right]
\end{aligned}$$

A subordinating conjunction may occur in a sentence medial position (subordinate clause follows the main clause as in ‘‘Du kannst den Wagen fahren, wenn du willst’’) or in a sentence initial position (the subordinate clause precedes the main clause as in ‘‘Wenn du willst, kannst du den Wagen fahren’’). These configurations are modelled by the first two entries. The last two entries account for recursive embedding of subordinate clauses, as in ‘‘Wenn... ,..., weil...’’; see Section 3.2.2.3 (p. 119) for further examples.

Adverb categories In main declarative clauses the Vorfeld must be non-empty. Consider the sentence ‘‘Der Mann schenkt seiner Frau jetzt einen Wagen’’ (*The man is giving his wife a car for a present now*). A subset of all word order variants of the sentence, including the unmarked syntax with the subject in the Vorfeld, are shown below:[5]

Der Mann	schenkt	seiner Frau jetzt einen Wagen
Seiner Frau	schenkt	der Mann jetzt einen Wagen
Einen Wagen	schenkt	der Mann jetzt seiner Frau
Jetzt	schenkt	der Mann seiner Frau einen Wagen
*Jetzt seiner Frau/einen Wagen	schenkt	der Mann einen Wagen/seiner Frau
*Seiner Frau/Einen Wagen jetzt	schenkt	der Mann einen Wagen/seiner Frau
*Jetzt der Mann	schenkt	seiner Frau einen Wagen
*Der Mann jetzt	schenkt	seiner Frau einen Wagen

⋮

The first four variants of the sentence are grammatically valid. Each of the three arguments of the ditransitive verb ‘‘schenken’’ (*give as a present*) as well as any optional adjunct can occupy the Vorfeld. More than one constituent in the Vorfeld (one or more verb arguments and a temporal adverb), as in the remaining variants, are not grammatically valid. The Rechte Klammer and the Nachfeld of the sentence remain empty.

In order to account for fronting elements other than verb arguments, the marking on the verb categories is complemented by a corresponding feature on the categories of word classes which can be fronted. The syntactic categories of adverbials, for instance, are set as follows:

$$\begin{aligned}\mathrm{ADV} := {} & \mathrm{S}\big[\,\mathit{VF} : +\,\big]\backslash \mathrm{S}\big[\,\mathit{VF} : +\,\big] \\ & \mathrm{S}\big[\,\mathit{VF} : +\,\big]/\mathrm{S}\big[\,\mathit{VF} : -\,\big]\end{aligned}$$

The first entry accounts for sentence medial and final adverb placement. The second entry accounts for adverbial fronting while ensuring that the finite verb immediately follows the fronted adverb. The unification mechanism guarantees that only those verb categories which are marked as $[\,\mathit{VF} : \text{-}\,]$ can combine with an adverb with the same marking, disallowing further fronted elements; see the third and fourth entries of the example category for the transitive ‘‘fuhr’’ (*drive*) on page 210.

[5] Ungrammatical sentences are marked as usual with an asterisk.

6.1.2 Mathematical expressions in the context of scope-bearing words

In order to account for interactions between symbolic mathematical expressions and natural language scope-bearing words, such as determiners, quantifiers, negation, etc., in their cotext, as illustrated with example (20) (p. 121), we identify salient structural parts of mathematical expressions that may be modified by natural language words which precede them. Each mathematical expression is reinterpreted in terms of these substructures by assigning them types of partial expressions. These categories are then combined with the surrounding linguistic context in the course of parsing.

Consider the example (20) reproduced below:

(79) B enthaelt kein $x \in A$

The expression $x \in A$, while in isolation has a surface form of a formula (truth-valued type), in the context of the sentence has the reading of a post-modified noun phrase “x which is in A” (object-denoting type). This is a systematic phenomenon involving scope-bearing modifiers in the left context of expressions of type FORMULA. Based on this observation, we obtain the intended reading by considering two systematically relevant salient substructures of mathematical expressions: the subexpressions directly below the top node in the expression's tree. (Recall the discussion in Chapter 3 Section 3.2.1.2 (p. 95) and Section 3.2.2.3 (p. 121).) For each expression of type FORMULA we produce two additional readings:

TERM _FORMULA where TERM denotes the expression left of the top-node operator and _FORMULA denotes the expression consisting of the top-node operator and the expressions to its right

FORMULA_ TERM where FORMULA_ denotes the expression consisting of the top-node operator and the expressions to its left and TERM denotes the expression right of the top-node operator

The underscore notation indicates an incomplete expression which requires material in the left (_FORMULA) or right context (FORMULA_). In the case of the expression $x \in A$, the two readings are TERM:=“x”, _FORMULA:=“$\in A$” and FORMULA_:=“$x \in$”, TERM:=“A”.

The corresponding syntactic categories for lexicon entries of mathematical expression types are:

TERM	:= NP
	N
FORMULA	:= S
_FORMULA	:= NP\NP

The category NP\NP is analogous to the resulting category of a restrictive (defining) relative clause and its semantics is ''which is _FORMULA'' (which could be also read as a ''such that''-clause: ''such that TERM is _FORMULA''). The corresponding category for FORMULA_ would be NP/NP, however, we did not find contexts in which partial expressions of this type would be relevant.

Each of the above readings is embedded within the original cotext in the course of preprocessing. (Recall the general architecture of the system presented in Section 5.2, p. 180):

TERM enthaelt kein FORMULA
TERM enthaelt kein TERM _FORMULA

Following this preprocessing, multiple readings of the sentence are interpreted (parsed). The first reading will fail because the category of ''kein'' (NP/NP) is not compatible with the category FORMULA (S), given prior type declarations, leaving the intended reading of (79) obtained through syntactic reinterpretation of the original formula.

6.1.3 Mathematical expression fragments

In order to account for mathematical expressions used as shorthand for natural language, as in (22), reproduced below,

(80) $A \cap B$ ist $\in$ von $C \cup (A \cap B)$
$A \cap B$ is $\in$ of $C \cup (A \cap B)$

both the mathematical expression and the natural language parser are adapted to support incomplete mathematical expressions and their interactions with the surrounding natural language text. To this end, the mathematical expression analysis process identifies incomplete expressions using knowledge of syntax and semantics of formal expressions in the given mathematical domain and assigns them symbolic tokens representing incomplete expression types.

In the case of (80), the mathematical expression parser identifies the symbol, $\in$, and, based on its knowledge of symbols in set theory, it finds that it is a formula-forming operator requiring two arguments: one of type `Inhabitant` and the other of type `Set`. The symbol is assigned a symbolic token _FORMULA_ and the utterance is preprocessed as:

TERM ist _FORMULA_ von TERM

In line with the lexicalised grammar approach, incomplete mathematical expressions as categories are modelled in the lexicon by compiling non-canonical constructions into the grammar; that is, symbolic tokens for incomplete expressions are included in the CG lexicon as pseudo-lexemes with appropriate syntatic categories. The entry for _FORMULA_ in the parser's lexicon for the occurrence above corresponds to the relational noun reading, ‘‘element (of)’’:

$$\text{\_FORMULA\_}_{\in} := \mathrm{NP/PP}\left[\, \mathit{lex} : \mathit{von} \,\right]$$

Other kinds of incomplete mathematical expressions and their types are treated in a similar way: by identifying their incomplete type (which is used as token during parsing) and introducing a corresponding entry in the parser's lexicon.

6.1.4 Irregular syntax

With utterance (23), reproduced below, we illustrated the use of domain-specific syntax while verbalising a formal expression in natural language:

(81) wenn A vereinigt C ein Durchschnitt von B vereinigt C ist, dann müssen alle A und B in C sein
If A union C is equal to intersection of B union C, then all A and B must be in C

The past participle ‘‘vereinigt’’ (*unified*) is normally used in a verbal prepositional construction: ‘‘vereinigen mit’’ + Dat. (*unify with*). The construction ‘‘A vereinigt B’’ is, however, commonly used in spoken verbalisation of the term $A \cup B$. (Recall the discussion on verbalisation of symbolic notation in Section 3.2.1.2, p. 100) In order to account for this kind of domain-specific constructions, appropriate syntactic categories for domain-specific lexemes are introduced into the parser's lexicon. In this case, the lexical entry for ‘‘vereinigt’’ includes a reading analogous to that of a mathematical operator, _TERM_, an incomplete term requiring terms to its left and right. The parser's lexicon includes the following syntactic category for the lexeme ‘‘vereinigt’’:

$$\text{vereinigt} := \mathrm{NP\backslash NP/NP}$$

Note that this category also enables parsing constructions such as ‘‘die Menge A vereining B’’ (*the set A union B*) with two readings: [[the set A] [union] [B]] and [[the set [A union B]]]. Of course, the lexicon also includes canonical categories for ‘‘vereinigt’’ as past participle.

6.2 Semantic phenomena

Out of the semantic phenomena illustrated in Section 3.2.2.4 (p. 124) we focus on ambiguity introduced by imprecise language and on computational reconstruction of the semantics of ‘‘the other way round’’. Imprecision of the kind we address here is frequently found not only in students' language, but also in mathematical textbooks, thus prioritising its modelling is well justified; see discussion in Section 3.2.2.4. The contextual operator is interesting because of its complexity and because a non-standard and non-trivial semantic procedure is needed to reconstruct its meaning. Moreover, to date, the literature on semantic and pragmatic factors in the use of ‘‘the other way round’’-like operators is scarce and there is little work on its computational modelling.

6.2.1 Imprecision and ambiguity

In Section 3.2.2.4 (p. 125 ff) we illustrated imprecise language which students use to refer to domain concepts precisely defined in mathematics; for instance, the subset relation is phrased using the verb ‘‘enthalten’’ (*contain*) (see example (20), p. 121) and the property of sets being disjoint is phrased using the word ‘‘verschieden’’ (*different*) (example (31), p. 126). Interpretation of imprecise and ambiguous language requires associating the linguistic meaning representations with plausible interpretations within mathematical domain. We model imprecise language in two stages: First, we extend the semantic lexicon with predicates which represent the semantics of imprecise, ambiguous, and informal expressions. Second, we represent the concepts in a domain ontology as generalisations of specific mathematical concepts. The linguistically-motivated domain ontology mediates between the lexical representations and domain interpretations. The two knowledge sources, outlined below, allow us to obtain the intended (possibly non-unique) domain-specific interpretation.

Semantic lexicon To mediate between the ambiguous linguistic realisations of domain concepts we use a semantic lexicon which maps the dependency frames generated by the parser to conceptual frames in a domain ontology (introduced further) or to interpretation scripts. The mapping is represented by means of rules. The input part of the rules are tuples defining tectogrammatical valency frames, that is, predicates and relations evoked by lexical items. The output structures are either the evoked concepts with roles indexed by tectogrammatical frame elements or interpretation scripts, that is, ‘‘recipes’’ for constructing symbolic meaning in the form of quantifier-free first order representations. Where relevant restrictions on role fillers – surface-lexical (marked with *lex*), lexico-semantic (*sem*), sortal (*type*), etc. – are specified.

Basic, most frequently used entries from the semantic lexicon were shown in Table 5.4 (Section 5.2.3.1, p. 198). Table 6.3 (p. 218) schematically shows further, more complex entries encoded in the lexicon for the most frequently recurring concepts relevant when talking about sets: *Containment* (set inclusion or membership), *Difference* (disjoint sets), and *Common property* (empty/non-empty intersection); see examples in Section 3.2.2.4 (p. 124). The symbols in bold are predicates with specific semantics, typewriter script denotes domain concepts from the domain ontology. (Notation is simplified. Technical information needed solely for implementation is omitted for readability.) The illustrated example entries are explained below.

Containment The *Containment* relation – (a) through (d) – is evoked by the predicate ‘‘enthalten’’ (*contain*) or by the *Location* relation. The tectogrammatical frame of ‘‘enthalten’’, (a) and (b), involves *Actor* and *Patient* dependents. Two entities are involved in *Containment*: *Container* and *Contents*. The former role is filled by the *Actor* dependent of the tectogrammatical frame and the latter by the *Patient* dependent. *Containment* is also evoked by the *Location* relation realised linguistically by a prepositional phrase with ‘‘in’’, (c), and involving the predicate ‘‘sein’’ (*be*) and the tectogrammatical relations *Actor* (as *Contents*) and *Location* (*Container*). Another realisation, (d), dual to the above, occurs with the adverbial phrase ‘‘außerhalb von (liegen/sein)’’ (*lie/be outside of*) and is defined as negation of *Containment*. In the domain ontology *Containment* specialises into the relations of (strict) `Subset` and `Membership`. A different kind of containment, (b), may be meant if the entities involved are interpreted merely in syntactic terms as mathematical expressions, as in ‘‘The term $A \cup B$ contains A’’ (a constructed example). In this case *Structural composition* is meant and the roles of the entities involved are those of a *Structured object* (a complex mathematical expression as the *Actor*), and a *Substructure* (a mathematical expression, complex or atomic, as *Patient*).

Difference The *Difference* relation – (e) and (f) – realised linguistically by the *HasProperty* TR with the predicative adjective ‘‘verschieden (sein)’’ (*be different*), involves a plural *Actor* (here: coordinated dependents (*Coord*)). A generalisation of this rule would involve an arbitrary number of coordinated entities and a matching number of *Object* arguments of *Difference*. This would also enable interpretation of ‘‘pairwise different’’ (a constructed example), for instance, by marking an attribute

Table 6.3: Example entries from the semantic lexicon; notation simplified, informal

TR structure	Lexical meaning
Containment	
(a) $(\textbf{contain}_{\text{PRED}}, \textit{Actor}_x, \textit{Patient}_y)$	$:= (\textit{Containment}, \textit{Container}_x, \textit{Contents}_y)$
(b) $(\textbf{contain}_{\text{PRED}}, \textit{Actor}_{x,\textit{type:complex-me}}, \textit{Patient}_{y,\textit{type:me}})$	$:= (\textit{Structural composition}, \textit{Structured object}\ _x, \textit{Substructure}\ _y)$
(c) $(\textbf{be}_{\text{PRED}}, \textit{Actor}_x, \textit{Location}_{y,\textit{lex:``in''}})$	$:= (\textit{Containment}, \textit{Container}_y, \textit{Contents}_x)$
(d) $(\textbf{be}_{\text{PRED}}, \textit{Actor}_x, \textit{Location}_{y,\textit{lex:``ausserhalb''}})$	$:= \textit{not}\ (\textit{Containment}, \textit{Container}_y, \textit{Contents}_x)$
Difference	
(e) $(\textbf{be}_{\text{PRED}}, \textit{Actor}_{\textit{coord}(x_1,\textit{type:Object};x_2,\textit{type:Object})}, \textit{HasProperty}_{\textit{lex:``verschieden''}})$	$:= (\textit{Difference}, \textit{Object}_{x_1}, \textit{Object}_{x_2})$
(f) $(\textbf{be}_{\text{PRED}}, \textit{Actor}_{\textit{coord}(x_1,\textit{type:Set};x_2,\textit{type:Set})}, \textit{HasProperty}_{\textit{lex:``disjunkt''}})$	$:= (e_1\ \texttt{Membership}\ x_1\ \textit{and}\ e_2\ \texttt{Membership}\ x_2 \Rightarrow e_1 \neq e_2)$
Common property	
(g) $(\textbf{have}_{\text{PRED}}, \textit{Actor}_{\textit{coord}(x_1,x_2,\ldots,x_n)}, (\textit{Patient}_{y,\textit{type:rel},\textbf{Pred}}, \textit{GenRel}_{\textit{lex:``gemeinsam''}}))$	$:= (\textbf{Pred}(x_1, y)\ \textit{and}\ \textbf{Pred}(x_2, y)\ \textit{and}\ \ldots\ \textit{and}\ \textbf{Pred}(x_n, y))$
(h) $(\textbf{have}_{\text{PRED},\textbf{Pred}}, \textit{Actor}_{\textit{coord}(x_1,x_2,\ldots,x_n)}, (\textit{Patient}_{y,\textit{type:non-rel}}, \textit{GenRel}_{\textit{lex:``gemeinsam''}}))$	$:= (\textbf{Pred}(x_1, y_k)\ \textit{and}\ \textbf{Pred}(x_2, y_k)\ \textit{and}\ \ldots\ \textit{and}\ \textbf{Pred}(x_n, y_k))$
(i) $(\textbf{Pred1}_{\text{PRED},\textit{type:rel}}, \textit{Actor}_{\textit{coord}(x_1,x_2,\ldots,x_n)}, (\textit{Patient}_{y,\textit{type:rel},\textbf{Pred2}}, \textit{GenRel}_{\textit{lex:``gemeinsam''}}))$	$:= (\textbf{Pred1}(x_1, y)\ \textit{and}\ \ldots\ \textit{and}\ \textbf{Pred1}(x_n, y)\ \textit{and}\ \textbf{Pred2}(x_1, y)\ \textit{and}\ \ldots\ \textit{and}\ \textbf{Pred2}(x_n, y))$

pairwise on the relation. The other kind of domain-specific difference, evoked by the domain term ‘‘disjunkt (sein)’’ (*(be) disjoint*) is analysed by means of an interpretation script which directly constructs the domain-specific interpretation.

Common property Having a ‘‘common property’’ – (g) through (i) – can be interpreted using three interpretation scripts. **Pred** here are meta-objects to be instantiated with the meaning of the TR node on which they are marked; PRED is a predicate head of a TR structure. The attributes *non-rel* and *rel* restrict instantiation to non-relational and relational predicates, respectively. The first entry, (g), models the case in which the *Patient* dependent is a relational noun and the TR predicate is **have**, as in one of the utterances in the corpus: ‘‘[A und B]$_{\mathit{Actor}:\mathit{coord}}$ haben [gemeinsame Elemente]$_{\mathit{Patient}:\mathit{rel},\mathbf{Pred}}$’’ (*$A$ and B have common elements*). The second entry, (h), is the case of a non-relational noun, as in ‘‘[Peter and Paul]$_{\mathit{Actor}:\mathit{coord}}$ [have]$_{\mathrm{PRED},\mathbf{Pred}}$ [a common car]$_{\mathit{Patient}:\mathit{non\text{-}rel}}$’’. The third, (i), covers the case of a relational noun and a relational predicate, as in ‘‘[Peter and Paul]$_{\mathit{Actor}:\mathit{coord}}$ [see]$_{\mathrm{PRED}:\mathit{rel},\mathbf{Pred1}}$ [a common friend]$_{\mathit{Patient}:\mathit{rel},\mathbf{Pred2}}$’’.

Linguistically-motivated domain ontology Domain-specific interpretations of concepts in the semantic lexicon are retrieved from a domain-ontology. Unlike the model in (Gruber and Olsen, 1994) our ontology is *linguistically-motivated*. It is a hierarchically-organised representation of objects, their properties, and types of property fillers, which serves as an intermediate representation mediating between imprecisely expressed concepts and a formal representation of knowledge for reasoning purposes. Horacek (2001b) and Horacek et al. (2004) motivate why this kind of representation is needed as an interface when mathematical knowledge is to be presented in natural language. Our representation is motivated by analogous phenomena on the language understanding side and, like the model in (Horacek et al., 2004), closely reflects knowledge representation in the domain reasoner, Ωmega.

In the *objects ontology* we model, among others, typographical properties of mathematical objects, including substructure delimiters (such as brackets), linear orderings (for instance, argument positions with respect to the head operator), and groupings or delimited substructures (for instance, bracketed subformulas). In the *relations ontology* we model imprecise relational concepts which have a meaning independent of the mathematical domain, but need to be interpreted in terms of their domain-specific meaning. Imprecisely expressed relations are modelled as general relations which subsume mathematical relations. The former provides access to substructures of mathematical expressions as potential antecedents of referring expressions (see Section 6.3.2). The purpose of the latter is to enable interpretation of ambiguous relations. For instance, in order to interpret an imprecise verb ‘‘enthalten’’ (*contain*), we model a relation `Containment` as a *semantic relation* in the ontology of relations. `Containment` holds between entities if one includes the other as a whole or if it includes components (elements) individually. This is a generalisation of the `(Strict) Subset` and `Membership` relations in set theory. An ambiguous lexical item ‘‘enthalten’’ is linked to the ambiguous concept which it evokes through the semantic lexicon and the concept is in turn given alternative domain-specific interpretations through the domain ontology; a basic example of how the meaning assignment is performed was shown in Section 5.3 (p. 200).

Excerpts from the ontologies of objects and relations are shown in Figures 6.2 (p. 221) and 6.3 (p. 222). Names of objects and relations are capitalised. Names of properties are in lower-case italics. (To simplify the presentation, certain constraints on fillers and links between properties are not shown.) Properties are inherited monotonically. Object specialisation in some cases introduces further properties (marked with ‘‘+’’) and in other cases, object properties become specialised (‘‘spec’’). For instance, the property *container* of the `Containment` relation is a more specific instance of the *argument* property of `Relation` propagated through `Semantic relation`. Value restrictions on properties are marked with ‘‘restr’’. Restrictions on number are marked with a number on a property. For instance, the filler of *right argument* (specialisation of *argument*) of `Set property` is restricted to be an object of type `Set` (in the objects ontology) and *left argument* of `Binary relation` must be unique (‘‘1’’).

The objects ontology includes moreover information on *mereological relations* between objects (not depicted in the figure for the sake of readability; we list examples below). Mereological relations concern both physical, surface properties of objects and ontological properties of objects. Part-of relations specific to our domain concern mathematical expression substructures (notation below is: *part-of(part, whole)*; not all objects mentioned here are shown in Figure 6.2):

part-of(`Subterm`, `Term`)
part-of(`Bracketed term`, `Term`)
part-of(`Term component`, `Term`)
part-of(`Subformula`, `Formula`)
part-of(`Bracketed formula`, `Formula`)
part-of(`Formula component`, `Formula`)

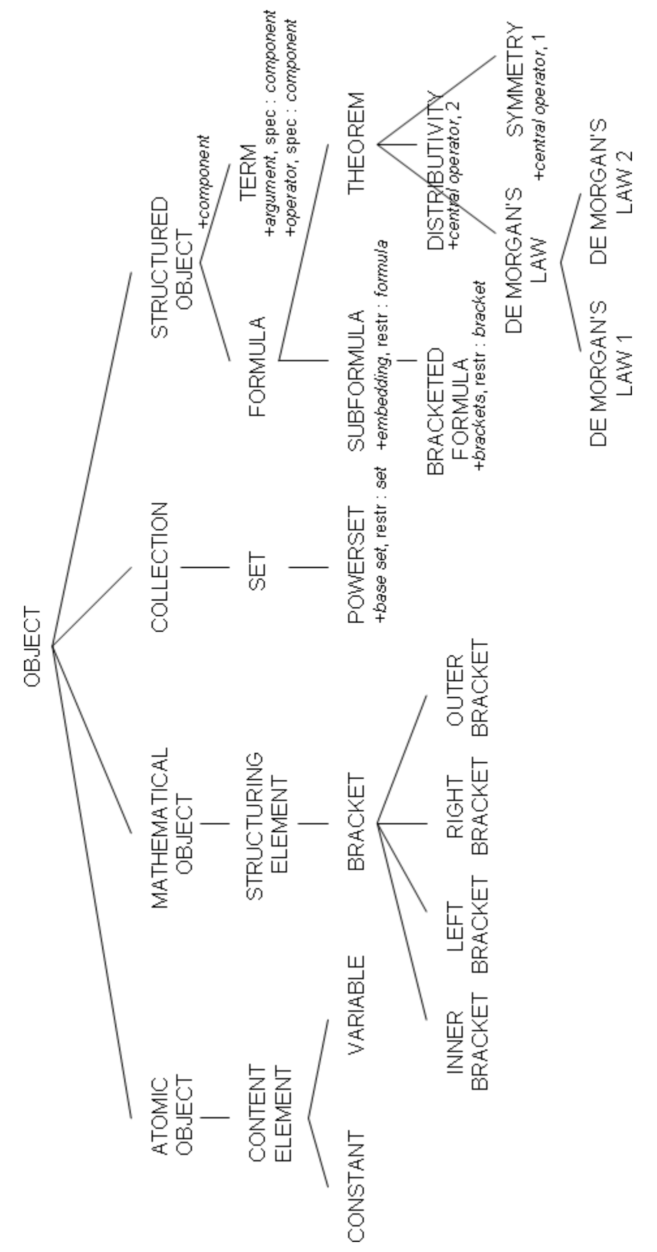


Figure 6.2: Excerpt of the representation of objects

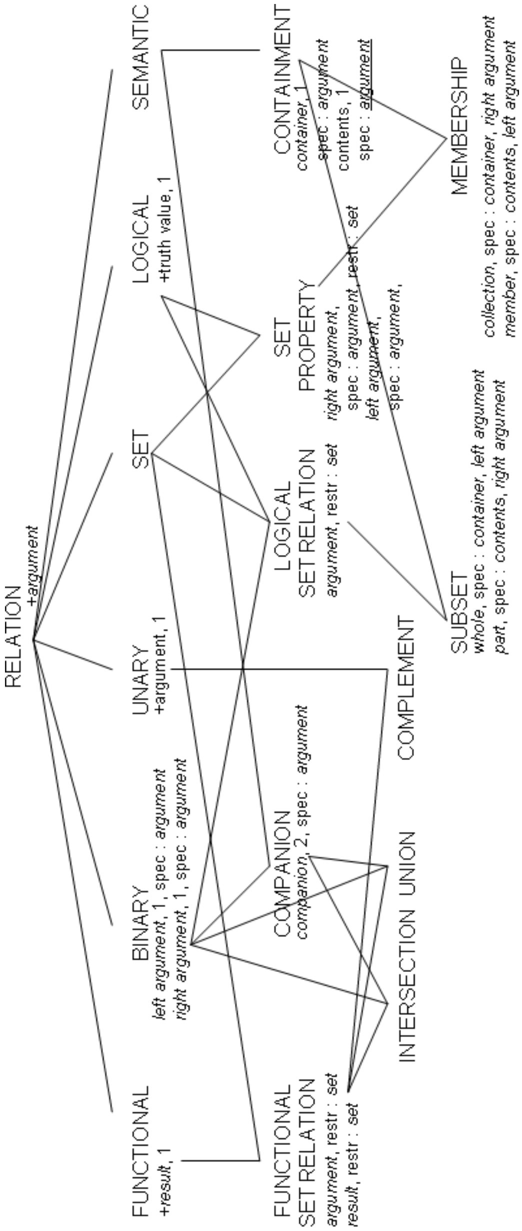


Figure 6.3: Excerpt of the representation of relations

These relations are especially relevant in resolving references to parts of notation (discussed in Section 3.2.2.5, p. 132; see also further in this chapter). Consider the commonly recurring fragment ‘‘Dann gilt für die linke Seite,...’’ (*Then for the left side it holds that...*). From the objects ontology we know that terms and formulas have sides:

property(`Structured object` $_{\texttt{Term}}$, *component*$_{side}$)
property(`Structured object` $_{\texttt{Formula}}$, *component*$_{side}$)

The predicate ‘‘gilt’’ (*hold*) in the context of a prepositional phrase with ‘‘für’’ (*for*) normally takes two arguments: one of type `Structured object` $_{\texttt{Formula}}$ (the formula that holds) and a PP argument of type `Structured object` $_{\texttt{Term}}$ or `Structured object` $_{\texttt{Formula}}$, rather than an argument which is a property (*side*). Using the objects ontology and the reinterpretation rule ‘‘object with property for property’’ (Section 6.3.2) we can obtain the intended interpretation.

6.2.2 “The other way round” semantics

‘‘The other way round’’ or the German ‘‘umgekehrt’’ is a complex operator of higher-order, that is, it takes a predicate or predicates as arguments. In the resulting proposition certain elements of the original proposition are ‘‘swapped’’, that is, the implicit proposition is a transformation of the verbalised proposition. Recall example (33) from C-I reproduced below:

(82) Wenn alle A in $K(B)$ enthalten sind und dies auch umgekehrt gilt, muß es sich um zwei identische Mengen handeln
If all A are contained in $K(B)$ and this also holds the other way round, these must be identical sets

In the above utterance, *the other way round* is ambiguous in that it may operate on immediate dependents of the verb ‘‘contain’’, resulting in the reading ‘‘all $K(B)$ are contained in A’’, or on its embedded dependents, yielding the reading ‘‘all B are contained in $K(A)$’’. The fact that the *Containment* relation is asymmetric and the overall task context – proving that ‘‘If $A \subseteq K(B)$, then $B \subseteq K(A)$’’ holds – suggest that the second interpretation is meant. (Similar other operators were discussed in Section 3.2.2.4, p. 128)

Human–human interaction frequently exploits the efficiency of implicitness in communication. By contrast, computational understanding of implicit semantics is non-trivial. Formal reconstruction of implicit meaning requires inference and resolving ambiguities, which, in turn, requires context understanding and domain knowledge in interpretation. Linguistic devices requiring *insertion* of omitted content, such as gapping and ellipsis, have been often addressed with computational approaches, however, there is virtually no work addressing structures whose reconstruction requires transformation, such as ''the other way round''. Chaves (2010) proposed an HPSG-based approach to modelling *vice versa*, however, evaluation was not performed. We studied contexts in which ''the other way round''-like lexemes occur in a collected corpus and devised an algorithm for resolving the implicit semantics. The reconstruction algorithm uses the deep semantic representations produced by the parser, transforms the semantic representations using patterns, and applies pragmatically- and empirically-motivated preferences to restrict the number of candidates. The reconstruction method is outlined in the following sections.

"The other way round" data

In order to learn about cross-linguistic regularities in ''the other way round'' constructions, we collected a corpus of German and English sentences in which they occurred. Aside from our tutorial dialogue data, the sentences stemmed from the Negra Frankfurter Rundschau corpora[6] and from the Europarl corpus (Koehn, 2005). The latter we used in a pilot evaluation. A subset of sentences stemmed also from internet searches. In all the data we searched for the German phrases ''andersrum'' and ''umgekehrt'', and their English equivalents ''the other way (a)round'' and ''vice versa''. Uses of ''umgekehrt'' as a discourse marker were excluded, as were the cases in which the transformation needed was of a lexical nature (such as finding an antonym) and instances of ''andersrum'' expressing a physical change (such as changing the orientation of an object; see, for instance, the use of ''umgekehrt'' in the Bielefeld corpus[7]). Example sentences are shown below:

(83) Technological developments influence the regulatory framework and vice versa.

(84) It discusses all modes of transport from the European Union to these third countries and vice versa.

(85) Ok – so the affix on the verb is the trigger and the NP is the target.
. . . No; the other way round

(86) Da traf Völler mit seinem Unterarm auf die Hüfte des für Glasgow Rangers spielenden Ukrainers, oder umgekehrt
Then Völler hit the hip of the Ukrainian playing for Glasgow Rangers with his lower arm, or the other way round

(87) Nowadays, a surgeon in Rome can operate on an ill patient – usually an elderly patient – in Finland or Belgium and vice versa.

[6] http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus[Accessed:2005]

[7] http://www.sfb360.uni-bielefeld.de[Accessed:2005]

(88) Der Ton der Klarinette ist wirklich ganz komplementär zu den Seiteninstrumenten und umgekehrt

The clarinet's tone is really very complimentary to strings and vice versa

(89) Wenn alle A in $K(B)$ enthalten sind und dies auch umgekehrt gilt, muß es sich um zwei identische Mengen handeln

If all A are contained in $K(B)$ and this also holds vice-versa*, these must be identical sets*

(90) Dann ist das Komplement von Menge A in Bezug auf B die Differenz $A/B = K(A)$ und umgekehrt

Then the complement of set A in relation to B is the difference $A/B = K(A)$ and vice versa

(91) Ein Dreieck mit zwei gleichlangen Seiten hat zwei gleichgroße Winkel und umgekehrt

A triangle with two sites of equal length has two angles of equal size, and vice versa

(92) . . . Klarinette für Saxophonist und umgekehrt

. . . a clarinet for a saxophonist and the other way round . . .

(93) Man muß hier das Gesetz der Distributivität von Durchschnitt über Vereinigung umgekehrt anwenden

One has to apply the law of distributivity of intersection over union in reverse direction

(94) Es gilt: $P(C \cup (A \cap B)) \subseteq P(C) \cup P(A \cap B)$. . . . . Nein, andersrum.

It holds: $P(C \cup (A \cap B)) \subseteq P(C) \cup P(A \cap B)$. . . . . No, the other way round.

(95) Wir wissen, daß sich Sprachen in Folge von geographischer Separierung auseinanderentwickeln, und nicht umgekehrt

We know that languages branch out as a result of geographical separation, not the other way round

Analysis of the examples reveals that ‘‘the other way round’’ appears in contexts which can be classified in terms of the type of elements which must be interchanged (‘‘swapped’’) in order to recover the implicit proposition. The four types of transformations needed to reconstruct the implicit semantics are summarised in Table 6.4 (p. 226).

Examples (83) through (86) illustrate the *Argument* swap. The transformation may be applied to different dependent roles, for instance, *Actor* and *Patient* dependents, as in (83), or *Direction-From/To* roles, as in (84). Transformation also works across clauses, as in (85). Example (86) shows that role fillers themselves may be complex structures and that their parts may participate in the transformation; in (86) world knowledge is needed in the reconstruction (obviously what is meant here are persons with their body parts and these together need to be swapped, not just the body parts or just the persons).

Table 6.4: Types of ''the other way round'' transformations

Transformation type	Description
Argument	Case role fillers (arguments) of a head need to be swapped (immediate daughters of a head)
Modifier	Argument modifiers swapped (lower dependents of a head)
Mixed	Combination of the two cases above (a modifier is swapped for an argument which, in turn, takes the role of the modifier in the reconstructed form)
Proposition	The proposition's ''dual'' needs to be applied (in some cases *Argument* swap can be applied there as well)

Modifier swap is shown in (87)–(88). Utterance (87) is ambiguous: from a structural point of view, it could be categorised as *Argument* swap, however, given world knowledge, this interpretation is rather infelicitous. A contextually-motivated metonymic reconstruction, prior to transformation, is required in (88); ''the strings'' needs to be interpreted as ''the tone of the strings''.

Mixed transformations are illustrated with utterances (89)–(92). The first example, (89), has been already discussed earlier in this section. In (90) multiple occurrences of the items need to be swapped and the transformation must be propagated to the formula. In (91) the properties of a triangle need to be swapped. This can be done based on the surface structure of the sentence. The resulting implication states that a triangle with two sides of equal length is a triangle with two equal angles. In this case, the reconstruction could also fall into the last type, *Proposition* transformation: reversing the implication. In (92), a lexical reinterpretation is needed prior to the reconstruction: ''a saxophonist'' needs to be expanded into ''a saxophone player'', so that the intended reading ''saxophone for a clarinet player (clarinetist)'' can be obtained.

Finally, examples (93)–(95), involve swapping entire *Propositions*; in the domain of mathematics, these may be formulas. In (93), the distributivity law needs to be applied ''right to left'' (rather than ''left to right'') and in (94), the superset relation needs to be swapped for subset. The last example, (95), requires structural recasting. Once the utterance's semantics is represented as headed by the *Result* TR, swapping the two propositions – ''branching out'' and ''becoming geographically separated'' – yields the desired result.

Processing "the other way round"

The examples show that ''the other way round'' transformation typically operates at the level of semantic roles of the elements in a sentence. Our last

Table 6.5: Examples of interchangeable relata in ‘‘the other way round’’ transformations; unless marked otherwise, *Relation*-dependents are involved

Interchangeable(Arg$_1$, Arg$_2$)
Interchangeable(*Actor*, *Patient*)
Interchangeable(*Direction-Where-From*, *Direction-Where-To*)
Interchangeable(*Time-From-When*, *Time-Till-When*)
Interchangeable(*Cause*-dependent, *Cause*-head)
Interchangeable(*Condition*-dependent, *Condition*-head)

category, *Proposition* transformation, can be in some cases also realised as an *Argument* transformation; for instance, instead of swapping $\supseteq$ for $\subseteq$ in (94), the two sides of the formula could be swapped. Clearly, however, the information relevant in meaning reconstruction is the sentence’s semantic dependency structure. In our approach we employ the tectogrammatical structure and show that it is an appropriate level of semantic description.

The linguistic analysis consists of semantic parsing, identification of candidate pairs whose elements are to be interchanged, followed by contextually-motivated reconstruction and optional recasting. In a fully automated setting, sentences would be analysed with a parser which constructs deep dependency-based representations of utterances’ linguistic meaning (as described in Section 5.2.3.1, p. 189) and which is integrated into a discourse processing architecture. Here we perform manual analysis.

Reconstruction heuristics Based on analysis of the corpora, we identified combinations of relations whose dependent arguments frequently participate in ‘‘the other way round’’ transformation. Examples of such relations are shown schematically in Table 6.5. Similarly to *Cause* and *Condition*, other discourse relation types of TRs can undergo head-dependent transformation (for instance, *Result*/*Effect*) or dependent-dependent transformations (enumerative relations, such as *Sequence* or *List* of the Rhetorical Structure Theory). During processing, we use the table of interchangeable relata as a preference criterion for selecting candidate relations for transformation. If one of the elements of a candidate pair is an *optional argument* which is not realised in the given sentence, we look at the preceding context to find the first instance of the missing element.

Reconstruction is performed based on formally defined rules for each of the identified transformation types shown in Table 6.4. The rules consist of a pattern part and an action part. Patterns are matched against the output of the semantic parser by identifying the relevant tectogrammatical roles and

Table 6.6: ‘‘The other way round’’ reconstruction rules

Transformation type	Reconstruction pattern
Argument	Pred(p) & $TR_1(p,x)$ & $TR_2(p,y)$ & Type-compatible(x,y) & Interchangeable(TR_1,TR_2) $\rightarrow$ Swap(p,x,y,p_t)
	Conj(p) & $TR_1(p,x)$ & $TR(x,u)$ & $TR_2(p,y)$ & $TR(y,v)$ $\rightarrow$ Swap(p,u,v,p_t)
Modifier	Pred(p) & $TR_1(p,x)$ & $TR^+_{11}(x,u)$ & $TR_2(p,y)$ & $TR^+_{21}(y,v)$ & $TR_1 \neq TR_2$ & Type-compatible(u,v) $\rightarrow$ Swap(p,u,v,p_t)
Mixed	Pred(p) & $TR_1(p,x)$ & $TR_{11}(x,u)$ & $TR_2(p,y)$ & $TR_1 \neq TR_2$ & Type-compatible(u,y) $\rightarrow$ Swap(p,u,y,p_t)
Proposition	Subord(p) & $TR_1(p,x)$ & $TR_2(p,y)$ & $TR_1 \neq TR_2$ $\rightarrow$ Swap(p,x,y,p_t)

accessing their fillers. Actions apply transformations (below) on the items identified by the pattern parts to build the implicit proposition.

The reconstruction rules are shown in Table 6.6.[8] There are two patterns for an *Argument* type transformation: If the scope of the swap is a single clause, two arguments (semantic roles) of compatible types are identified as interchangeable. For the case of a two-clause scope, the relation must be a conjunction and swapped are arguments in the same relations. In a *Modifier* swap, type compatible modifiers of distinct arguments are selected. For a *Mixed* swap, a dependent is selected, as in the first case of *Argument* swap, and a type-compatible modifier of another argument, as in a *Modifier* swap. *Proposition* swap has the effect of inverting two clauses.

Rules are applied to the parser output (see Section 5.2.3.1, p. 189). For each node p, all patterns are matched with the node's dependency substructure and, if successful, the result is bound to p_t (transformed). Pred(p) is a function which checks if p has a PRED feature, that is, it is a proposition. Similarly, Conj(p) and Subord(p) test if a node is a complex proposition, coordination or subordination, respectively, based on a list of TRs denoting complex syntactic structures. Within a structure, dependents (participants and modifiers) in specific tectogrammatical roles are accessed by the function TR(p,x), where x specifies the TR-dependent of p; subscripts on x define constraints on the relations. TR$^+$ is a generalisation of TR which covers iterative embeddings (multiple occurrences of a *TR*; roles in the chain are not required to be identical). Aside from access functions, two functions test constraints on the identified

[8] Rules and the algorithm in this section are presented in an informal-schematic way.

Table 6.7: Recasting rules for ‘‘the other way round’’ reconstruction

Rule	Formalisation
Lexical recasting (lexical expansion)	Pred(p) & $\mathit{TR}_1(p,x)$ & Lex-Expand(x, u, *TR*, v) & $\mathit{TR}_2(p,y)$ & $\mathit{TR}_1 \neq \mathit{TR}_2$ & Type-compatible(v,y) $\rightarrow$ Swap(p, x, *TR*(u, v),p_t) & Swap(p_t, y, v, p_t)
Role recasting (optional role as head of an obligatory role)	Pred(p) & $\mathit{TR}_1(p,u)$ & $\mathit{TR}_2(p,v)$ & Type(u, t_u) & Type(v, t_v) & Recastable(TR_2, t_v, TR_3, t_u) & $\mathit{TR}_3(p,w)$ & Type-compatible(v, w) & $\mathit{TR}_1 \neq \mathit{TR}_2$ & $\mathit{TR}_1 \neq \mathit{TR}_3$ & $\mathit{TR}_2 \neq \mathit{TR}_3$ $\rightarrow$ Swap(p, u, v, p_t) & Add(p_t, $\mathit{TR}_3(v,u)$) & Remove(p_t, TR_2)
Proposition recasting (optional role as a discourse relation)	Pred(p) & $\mathit{TR}(p,x)$ & Member(*TR*, Subords) $\rightarrow$ Build((p_{TR}, $\mathit{TR}_1(p,y)$, $\mathit{TR}_2(p,$ Remove(p, TR))))

items: Interchangeable(TR_1, TR_2) tests whether a pair of relations is a good candidate for a transformation, based on the table of interchangeable relations (examples in Table 6.5). Type-compatible(x, y) tests whether the types of x and y are compatible according to an underlying domain ontology. In the case of proofs, this is an ontology of mathematical objects.[9] The action part of the patterns is realised by Swap(p, x, y, p_t) which replaces all occurrences of x in p by y and vice versa, and binds the result to p_t. Different applications of this operation result in different instantiations of x and y with respect to p.

In addition to pattern matching tests, candidates for *Argument* and *Proposition* transformations undergo a feasibility test to check if the predicate (PRED) whose roles would be swapped is known to be symmetric or asymmetric. If it is asymmetric, the result is implausible for semantic reasons. If it is symmetric, for pragmatic reasons (the converse proposition conveys no new information). In both cases a swapping operation is not performed.

Finally, a set of *recasting rules* is invoked to reorganise semantic representations prior to testing applicability of reconstruction rules. Recasting operations use additional test functions: Lex-Expand(x, y, *TR*, u) expresses the semantics of x by y with u in a *TR* relation. Type(x, t) associates the type t with x. Type is used to access a table of recastables roles. Recastable(TR_1,t_1,TR_2,t_2) verifies whether TR_1 with filler of type t_1 can be expressed as TR_2 with filler t_2, Add(p, a) expands p by an argument a, Remove(p, x) removes substructure x of p, and Build(s) creates a new structure s.

Three recasting rules, shown in Table 6.7, are defined: *Lexical recasting* performs lexical expansions of lexemes in order to accommodate the fact

[9] We did not construct a large-scale ontology of mathematical objects. In an automated system such a knowledge source would be of course necessary. For the purpose of the evaluation here we assume that such knowledge base exists.

```
$Scopes \leftarrow \epsilon$, $Structures \leftarrow \epsilon$, TR Structure $\leftarrow$ Parse $s$

// Identify scopes for swapping
if Pred($p$) then
  | $Scopes \leftarrow \{p\}$
end if
foreach (Subord(x) $\vee$ Conj(x)) $\wedge$ $TR(x, z)$ do
  | $Scopes \leftarrow Scopes \cup \{x, z\}$
end foreach

// Match patterns to build swapped structures
foreach $Scope$ in $Scopes$ do
  | $Structures \leftarrow Structures \cup$ X-Swap($Scope$) $\cup$ X-Swap(Y-Recast($Scope$))
end foreach
return Sort(Rank($Structures$))
```

Figure 6.4: Pseudo-code of ‘‘the other way round’’ reconstruction algorithm

that semantics of some lexemes conflates the meaning of two related items. Lexical representations are expanded if there is a sister role with a filler whose type is compatible with the type of the expanded item. *Role recasting* is performed if a dependent appears as a sister node in an overarching TR, that is, if functional dependency is not reflected by linguistic dependency; the dependent filler is removed and inserted as a modifier of the item on which it is dependent. *Proposition recasting* is performed if a proposition in a subordinating (discourse) relation is expressed as a TR: the argument (TR dependent) is lifted and the discourse relation is expressed as multiple-relation structure (consider the structure transformation needed to cover example (95)).

Reconstruction algorithm The structure building algorithm consists of two steps. First, the scope for applying the heuristics defined in Table 6.6 (p. 228) is determined and, second, results of rule matching are collected. For practical reasons, presently we make a simplifying assumption concerning the scope of the operator: While the effect of ‘‘the other way round’’ may range over entire paragraphs, we only consider single sentences with at most two coordinated clauses or one subordinated clause. This restriction is plausible for application-oriented systems; only a few examples from the corpora we have examined cannot be handled due to this simplification.

The algorithm is summarised in Figure 6.4. It takes an input sentence s, parses it, analyses its dependency structure to find predicate nodes, and binds potential dependency substructure scopes to the variable $Scopes$. For complex sentences, the structure(s) in each clause are also potential scopes. Next, each transformation rule (Table 6.6) is tested against the candidate scopes and the results are collected in $Structures$. The function X-Swap($Scope$) builds all instantiations of a given rule applied to $Scope$: X stands for *Argument*, *Modifier*, *Mixed*, and *Proposition*. Alternative parameters stemming from recasting (Table 6.7) are invoked with X-Swap(Y-Recast($Scope$)), where Y is *Lexical*, *Role*, or *Proposition recast* provided that they fit the pattern. If multiple readings are generated, they are ranked according to the following ordered criteria: (1) the nearest scope is preferred, (2) operations which swap ‘‘duals’’, such as left–right, are ranked higher, (3) constructed candidate phrases are matched against a corpus; pairs with higher bigram frequencies are preferred.

The linguistic analysis, the structure reconstruction patterns, the recasting rules, and the algorithms operating on top of these structures are formulated in a domain-independent way, also ensuring that the tasks involved are clearly separated. It is thus up to the concrete application to elaborate the required lexical semantic definitions (for instance, the lexical expansion for ‘‘saxophonist’’ in (92) to capture the example), to define the tables Interchangeable and Recastable, and to adjust the preference criteria.

Preliminary evaluation

A preliminary evaluation of the reconstruction algorithm has been performed on a sample of English and German sentences from Europarl (Koehn, 2005). Since we do not have access to a wide-coverage semantic dependency parser for English and German, manual evaluation has been conducted.

Evaluation data The evaluation set was created by extracting sentences from Europarl using the following regular expression patterns: (i) for English: phrases ‘‘the other way a?round’’ or ‘‘vice[-]?versa’’[10] (ii) for German: (ii-a) the word ‘‘umgekehrt’’ preceded by a sequence of ‘‘und’’ (*and*), ‘‘oder’’ (*or*), ‘‘sondern’’ (*but*; in the sense of *instead*), ‘‘aber’’ (*but*) or comma, optional one or two tokens and optional ‘‘nicht’’ (*not*), (ii-b) the word ‘‘umgekehrt’’ preceded by a sequence ‘‘gilt’’ (*holds*) and one or two optional tokens, (ii-c): the word ‘‘anders(he)*rum’’. 137 sentences have been retrieved using these criteria. Given the present limitation of the algorithm, we manually excluded those sentences whose interpretation involved the preceding sentence or paragraph,[11] as well as those in which the interpretation was explicitly spelled out. There were 27 such instances. The final evaluation set consisted

[10] The question mark denotes an optional element.

[11] For example: ‘‘Mr President, concerning Amendment No 25, I think the text needs to be looked at because in the original it is the other way round to how it appears in the English text.’’

Table 6.8: Distribution of transformation patterns in ‘‘the other way round’’ test data

Transformation type	No. of instances
Argument	64
Modifier	5
Argument/Modifier	3
Mixed	6
Argument/Mixed	2
Proposition	1
Argument/Proposition	1
Lexical	18
Other	10

of 110 sentences: 82 sentences in English–German pairs and 28 German-only. The reason for this difference is that the English equivalents of the German sentences containing the word ‘‘umgekehrt’’ may contain phrases other than ‘‘the other way round’’ or ‘‘vice versa’’. Depending on context, phrases such as ‘‘conversely’’, ‘‘in reverse’’ or ‘‘the reverse’’, ‘‘the opposite’’, ‘‘on the contrary’’ may be used. Here, we targeted only ‘‘the other way round’’ and ‘‘vice versa’’ phrases. If the German translation contained the word ‘‘umgekehrt’’, and the English source one of the alternatives to our target, only the German sentence was included in the evaluation. Because the distribution of sentences between the two languages is to a large degree unbalanced, cumulative results for both languages are reported.

Distribution of categories The structures in the evaluation set have been manually categorised into one of the transformation types from Table 6.4 and the elements of the dependency structures participating in the transformation have been marked.[12] Table 6.8 shows the distribution of transformation types in the data set. Counts for alternative interpretations are included. For instance, *Argument/Modifier* means that either *Argument* or *Modifier* transformation can be applied with the same effect; as in ‘‘External policy has become internal policy, and vice versa’’: either ‘‘external’’ and ‘‘internal’’ may be swapped (*Modifier*) or the whole NPs ‘‘external policy’’ and ‘‘internal policy’’ (*Argument*). *Lexical* transformation means that none of the rules was applicable; a lexical paraphrase (such as use of an antonym) needed to be performed in order to reconstruct the underlying semantics (that is, no structural transformation

[12] The author of this thesis annotated half of the data set. The other half was annotated by the collaborator in this work (see (Horacek and Wolska, 2007)), Dr Helmut Horacek.

Table 6.9: Evaluation results of ‘‘the other way round’’ transformations

Transformation type	Evaluation category			
	Correct	Ambiguous	Wrong	Failed
Argument	46	17	0	1
Modifier	3	2	0	0
Argument/Modifier	3	–	0	0
Mixed	4	2	0	0
Argument/Mixed	2	–	0	0
Proposition	1	0	0	0
Argument/Proposition	0	–	0	1
Lexical	16	0	2	0
Other	8	0	2	0

was involved). *Other* means that a transformation-based reconstruction was involved, however, none of our rules covered the structure.

Results Transformation results have been manually classified into four categories: ‘‘Correct’’ means that the algorithm returned the intended reading as a unique interpretation (this includes correct identification of lexical paraphrases (the category *Lexical* in Table 6.8), ‘‘Ambiguous’’ means that multiple results were returned with the intended reading among them, ‘‘Wrong’’ means that the algorithm returned a wrong result or, if multiple results were found, the intended reading was not included; ‘‘Failed’’ means that the algorithm failed to find a structure to transform because none of the rules matched.

Evaluation results are shown in Table 6.9. Complete corpora from which our data stemmed were used to build bigram frequencies. In case of possible alternative assignments (as in *Argument/Modifier*) Correct was assigned whenever the algorithm selected one of the possible assignments, independently of which one it was. The Correct results for *Other* are trivial: the algorithm correctly identified the 8 cases to which no rule applied. The two Wrong results for *Other* mean that a pattern was identified, but not the intended one. In two cases, the algorithm failed to identify a pattern even though a structure exhibited a pattern in one of the known categories (*Argument* and *Proposition*) (false negatives).

Discussion The most frequently occurring pattern in our sample is *Argument*. This is often a plausible reading. However, in 3 of the 4 false positives (Wrong results), the resolved incorrect structure was *Argument*. A baseline consisting of always assigning the most frequent category, *Argument*, would miss the other categories (altogether 12 instances) and yield the final result of 63 Correct (as opposed to 96; after collapsing the Correct and Ambiguous categories) and 15 (as opposed to 4) Wrong assignments.

The two missed known categories (Failed) involved multiple arguments of the main head: a modal modifier of the predicate (modal verb) and an additive particle (''also'') in one case, and rephrasing after transformation in the other case. To improve performance on cases such as the former, a list of dependents which the transformation should exclude as candidates could be incorporated into the algorithm. Among the patterns we did not anticipate, we found four types (one instance of each in the sample) which can potentially frequently recur: aim and recipient constructions involving a predicate and its *Aim* and *Beneficiary* dependent respectively, a temporal-sequence in which the order of the sequence elements is reversed, and a comparative structure with swapped relata. The remaining 6 structures require a more involved procedure: either the target dependent is deeply embedded or paraphrasing as well as morphological transformation of the lexemes is required. Overall however, the presented algorithm is a good first step towards automated reconstruction of the operator's semantics.

6.3 Reference phenomena

Computational approaches to anaphor resolution (or (co-)reference resolution more generally) typically address narrative text genres and use manually handcrafted rules, machine learning, or a combination of both to find antecedents. Syntactic, semantic, and lexical features of the anaphor carrier sentences and of the sentences containing candidate antecedents as well as probabilistic distributional properties of the anaphor in context are used as indicators of coreference; see, for instance, (Botley et al., 1996; Mitkov, 2000; Poesio et al., 2010) for an overview on reference resolutions algorithms. Anaphor resolution in dialogue have been gaining attention, however, reference resolution in dialogue proves more difficult and the performance of algorithms on dialogue corpora tends to be lower than on narrative discourse corpora (Poesio et al., 2010). Recently also studies specific to tutorial dialogue have been conducted; see, for instance, (Pappuswamy et al., 2005; Poesio et al., 2006).

A peculiarity of mathematical discourse is that referring expressions in this domain may be used to refer to the elements of formal notation. Examples of such references were shown in Section 3.2.2.5 (p. 132). References may address entire formal expressions or their parts. Most frequent are references to propositions, specifically, proof steps, verbalised in natural or in the symbolic language. Table 6.10 shows the distribution of references to object-denoting

Table 6.10: References to object-denoting terms and proof steps

Antecedent type	Data set		
	C-I	C-II	C-I&C-II
Object-denoting term	26	13	39
Proof step	35	81	116
Column totals	61	94	155

terms expressed symbolically (parts of mathematical notation) and to proof steps (expressed using mathematical notation or natural language) in the student turns in our corpora. (The types of referential forms included in this summary will be elaborated in the next section.) Overall, the number of occurrences of referring expressions is small (155 instances). Assuming one referring expression per turn, only around 12% of all student turns contain referring expressions to terms or proof steps (there are 1259 turns in total; see Table 4.1 on p. 161). There are more referring expressions in C-II (94) than in C-I (61), however, considering that C-II contains almost three times as many student turns as C-I, there are proportionally more referring expressions in C-I (on average, around 18% student turns with a referring expression in C-I and 10% in C-II). In the case of the tutorial dialogue scenario, antecedents of referring expressions may be found in either student's or tutor's turns. In spite of a seemingly high potential for ambiguity (many candidate symbolic terms as antecedents), in our experiments only in one case did the tutor initiate an explicit subdialogue to clarify a student's ambiguous use of reference.

In the following sections we look more closely into two aspects of modelling reference phenomena in proof tutoring dialogues. First, we conduct a corpus study on the types of referring expressions. Anaphor resolution algorithms are typically tailored to resolving expressions of a specific form, for instance, pronominal anaphora or references to expressions of specific type, for instance, discourse deictic anaphors (as in (Pappuswamy et al., 2005)). It is therefore useful to know what types of anaphora occur most frequently in our genre and to what entity types they refer. Second, we analyse the referring expressions in terms of their discourse scope. Considering the low overall number of instances of referring expressions in our corpora and especially the low number of object-denoting references, we do not propose a complete computational reference resolution algorithm. More data would need to be collected in order for a plausible computational algorithm to be developed. Instead, we again analyse the corpus data with respect to the location of the different antecedent types. The analysis of referential scope is relevant in determining the discourse scope for antecedent search, thus the two corpus-based analyses form a good basis for a computational algorithm to be developed and evaluated once more data is available. Finally, we show how the interpretation resources need to be extended in order to address indirect references specific to proof discourse.

6.3.1 Forms of referring expressions and scope of reference

Linguistic referring devices in the students' utterances include pronouns, pronominal and locative adverbs, noun phrases, demonstratives (discourse deixis), and definite articles. As will be shown further, all of them have been used to refer to parts of symbolic notation as well as to propositions or partial proofs (sequences of propositions) constructed in the course of dialogue. Examples of the different referring expression types are shown below.

Pronouns The following examples illustrate the use of pronominal anaphora:

(96) Da, wenn $A \subseteq K(B)$ sein soll, A Element von $K(B)$ sein muss. Und wenn $B_i \subseteq K(A)$ sein soll, muss es$_i$ auch Element von $K(A)$ sein.
Because if $A \subseteq K(B)$ should hold, A must be an element of $K(B)$. And if $B \subseteq K(A)$ should hold, B must be also an element of $K(A)$.

(97) T19: Erinnern Sie sich daran, [dass es ein z gibt mit $(x, z) \in S^{-1}$ und $(z, y) \in R^{-1}$.]$_i$
Do you remember that there is a z such that $(x, z) \in S^{-1}$ and $(z, y) \in R^{-1}$
S14: Ja, ich habe es$_i$ vorausgesetzt
Yes, it was an assumption I made

In (96) a personal pronoun, ‘‘es’’ (*it*), is used to refer to a term. The term is part of a formula and its syntactic/semantic function in the formula can be viewed as that of a subject/agent, parallel to the semantic function of the anaphor. The reference is local; the antecedent is in the same turn. Notice that it is hard to produce an comparable structure in English. The reference in German works because the formula is again used as shorthand for natural language; the subordinate clause reads ‘‘wenn B Teilmenge von $K(A)$ sein soll’’ and the pronoun refers to its subject. (In the given task context, this is the more plausible interpretation. An alternative antecedent candidate could be A and considering the student's confusion about the set membership and subset relations, it is not impossible that he actually meant to refer to A.) The pronoun in (97) is referring to the proposition in the preceding tutor's turn T19, that is, the antecedent is found in the other speaker's turn.

Pronominal and locative adverbs Pronominal adverbs (adverbial pronouns; ‘‘präpositional pronomen’’) are lexical constructions in Germanic languages formed by joining a preposition with a pronoun. Their anaphoric character is due to the pronoun obtaining thereby a locative adverb function. English examples include ‘‘thereby’’ (*by this*) or ‘‘therefor(e)’’ (*for that*) and German ‘‘damit’’ (*with that*) or ‘‘dafür’’ (*for that*). Locative adverbs in mathematical discourse also have an anaphoric character; consider, for instance, the frequent scope bearing locative ‘‘hence’’ in English. The dialogue fragments below illustrate the use of anaphoric adverbs in our corpora:

(98) S2: $[\, R \circ S\,]_i := \{(x, y) \mid \exists\, z(z \in M \wedge (x, z) \in R \wedge (z, y) \in S)\}$

. . .

S3: Nun will ich das Inverse davon$_i$
Now I want the inverse of it

(99) S7: Also [[ist $(z, x) \in S$ und $(y, z) \in R$]$_i$ und damit$_i$ auch [$(y, x) \in R \circ S$]$_j$]$_k$
Therefore it holds that $(z, x) \in S$ and $(y, z) \in R$ and by that also $(y, x) \in R \circ S$

. . .

S8: Somit$_{j?k?}$ ist $(x, y) \in (R \circ S)^{-1}$
With this it holds that $(x, y) \in (R \circ S)^{-1}$

In (98), a pronominal adverb ‘‘davon’’ (*of it*) is used to refer to a complex term, $R \circ S$, on the left-hand side of the definition. In principle, the reference is ambiguous: a competing antecedent for ‘‘davon’’ is the definiens part. In (99) the adverbial pronoun ‘‘damit’’ (*with this*) in S7, refers to the proposition in the first clause of the utterance. The pronominal adverb ‘‘somit’’ (*with that*) in S8 in the same excerpt may refer to the conjunction or implication of the assertions in S7 (marked with k) or only to the last assertion (marked with j).

Noun phrases Within this category we consider referential uses of noun phrases including deictic NPs, such as ‘‘(in) dieser Menge’’ (*(in) this set*) referring to a set expression in the dialogue fragment (56), reproduced below:

(100) S33: Nach Aufgabe W ist $(S \circ (S \cup R)^{-1})^{-1} =$ [$((S \cup R)^{-1})^{-1} \circ S^{-1}$]$_i$
By Exercise W: ... holds

. . .

S34: Dies$_i$ ist nach Theorem 1 gleich [$(S \cup R) \circ S^{-1}$]$_j$
This is by Theorem 1 equal to $(S \cup R) \circ S^{-1}$

. . .

S35: Ein Element (a, b) ist genau dann in [dieser Menge]$_j$, wenn ...
An element (a, b) is in this set if and only if ...

Definite noun phrases used to refer to elements of mathematical notation often involve metonymic reinterpretation. In Section 3.2.2.5 we already showed examples such as ‘‘die innere Klammer’’ (*the inner parenthesis*), ‘‘die linke Seite’’ (*the left side*) or ‘‘beide Komplemente’’ (*both complements*) (see p. 132). These are indirect references to structural parts of mathematical expressions, terms in formulas; ‘‘the left side’’ refers to the term to the left of the top-node operator in a formula, ‘‘the inner parenthesis’’ to a bracketed subterm of a bracketed term in a formula (rather than to a bracket itself), and the quantified noun phrase, ‘‘both complements’’ in ‘‘de morgan regel 2 auf beide komplemente angewendet’’ (*de morgan rule 2 applied to both complements*) to two terms headed by the complement operator.

Both definite and bare noun phrases can be also used generically to refer to concepts in the domain, for instance, to the concept of the set union as in: ‘‘The union of sets R and S contains all elements from R and all elements from S’’ (example (43), p 133) or ‘‘Potenzmenge enthaelt alle Teilmengen, also auch $(A \cap B)$’’ (*Powerset contains all subsets therefore also* $(A \cap B)$). In the latter case, ‘‘powerset’’ is a generic reference, whereas ‘‘$(A \cap B)$’’ is a specific reference to a subset of a specific instance of a power set introduced earlier. Moreover, named theorems and lemmata may be referred to by their proper names, for example, ‘‘de Morgan’s rule 2’’. These non-anaphoric uses are not included in further analyses.

Demonstratives The last type of referring expressions we analysed were deictic references by means of demonstrative pronouns, as in:

(101) Wenn [alle A in $K(B)$ enthalten sind $]_i$ und dies$_i$ auch umgekehrt gilt, muß es sich um zwei identische Mengen handeln
If all A are contained in $K(B)$ *and this also holds the other way round, these must be identical sets*

where the demonstrative pronoun ‘‘dies’’ (*this*) refers to a preceding proposition, or as in the previous example, (100), where ‘‘dies’’ in S34 refers to the term on the right-hand side of the formula in S33.

As a preliminary stage towards developing an anaphor resolution algorithm we conducted two studies on reference phenomena: First, we looked at the frequency of use of the illustrated forms to refer to entities specific to mathematics: domain objects evoked using symbolic notation and proof steps expressed in natural language or using symbolic expressions. Next, we looked at the discourse-referential scope of the expressions, that is, the scope of discourse, with respect to the referents, within which an antecedent is found.

Instances of anaphoric references and their antecedents have been annotated in the two corpora by the author. Discourse was interpreted cooperatively,

Table 6.11: Distribution of referring expression types by antecedent type

Antecedent type	Form of referring expression			
	Pronominal or Locative adverb	Noun phrase	Demonstrative	Pronoun
Object-denoting term	2	30	2	5
Atomic	0	2	0	2
Complex	2	28	2	3
Proof step	59	15	40	2
ME	28	10	27	0
ME & NL or NL	31	5	13	2
Column totals	61	45	42	7

in the sense that the most plausible candidate was considered as the antecedent, even if students' statements were invalid or incomplete. Multiple annotations have not been performed for the same reason as in Section 4.1: antecedent annotation does not require linguistic knowledge, but rather knowledge of the mathematical domains and understanding of the solution constructed in the course of dialogue. Considering the fact that the set theory and binary relations proofs are of low complexity, the most plausible antecedent types could be identified by cooperatively interpreting the students' intentions and taking into account the information about the student obtained in the course of dialogue. Referential scope may be ambiguous in the case of references in invalid steps or incomplete proofs (omitted steps). In case of uncertainty, we annotated the turn in which the first plausible candidate was found.

Table 6.11 shows the distribution of referring expression types to two types of entities: object-denoting terms and proof steps. Further distinction is made between atomic and complex terms (as in A and $A \cup B$, respectively) and proof steps expressed in the symbolic notation (ME category; see Section 4.3.1, p. 160) or using some natural language (ME & NL and NL categories). The largest class of referential forms are pronominal and locative adverbs, the majority of which refer to proof steps (or larger parts of proofs). There are approximately the same number of nominal references as deictic references using demonstratives, however, there are clear differences in their use: the former are mainly used to refer to parts of notation (object-denoting terms), while the latter are mainly used to refer to proof steps. Further analysis of the dialogues revealed that the majority of the latter types occur in chaining equation contexts in which a formula is contributed and the next rewriting step is introduced by phrasing ''This is then (equal to)...'' or analogous. The majority of nominal references to terms are indirect references of the kind discussed in Section 3.2.2.5 (p. 134). The number of pronominal anaphora is

Table 6.12: Distribution of reference types by the location of the antecedent

Antec. type	Form of referring expression	Location of the antecedent					
		Same turn	S_{-1}	T_{-1}	$S_{\geq -2}$	$T_{\geq -2}$	Task descr.
Object-denoting term	Pronominal or locative adverb	1	1	0	0	0	0
	Noun phrase	4	5	4	9	8	10
	Demonstrative	0	2	0	0	0	0
	Pronoun	3	0	0	2	0	0
	Subtotals	8	8	4	11	8	10
Proof step	Pronominal or locative adverb	21	38	0	0	0	0
	Noun phrase	4	6	0	2	3	3
	Demonstrative	22	16	2	0	0	0
	Pronoun	0	1	1	0	0	0
	Subtotals	47	61	3	2	3	3
	Column totals	55	69	7	13	11	13
	(% all references)	(35)	(45)	(5)	(8)	(7)	(8)

surprisingly small; only 7 occurrences overall. In all cases of ‘‘es’’-references (neuter personal pronouns) to object-denoting terms, the anaphor was the entity on the left side of a mathematical expression of type formula. The low number of pronominal references to terms can be perhaps explained by the fact that nominal reference is more specific and thus reduces the chance of unintended interpretation; compare referring to a left-hand side of an equation with ‘‘die linke Seite’’ vs. ‘‘es’’, as in (96): while there may be multiple ‘‘left sides’’ competing as candidates, the structure of the expressions which embed them is a good cue in resolution; recall the discussion in Section 3.2.1.2 (p. 95).

Table 6.12 shows the distribution of reference types by the location of the antecedent. The interpretation of columns is the following: ‘‘Same turn’’ means that the antecedent is found in the same turn as the referring expression (as in the example (96)), ‘‘S_{-1}’’ and ‘‘T_{-1}’’ mean that the antecedent is in the preceding student or tutor turn, respectively (as in (97) and (98)), ‘‘$S_{\geq -2}$’’ and ‘‘$T_{\geq -2}$’’ mean that the antecedent is in a student or tutor turn, two or more turns prior to the anaphor, ‘‘Task descr.’’ (task description) means that the antecedent is in the first tutor turn which specifies the proof task. (Note that the task may have been specified in the immediately preceding turn if the analysed turn is the student’s first contribution.) What can be seen from the annotation results is that the majority of the references to proof steps are local, whereas references to terms may have a large scope. Out of the 39 references to object-denoting terms, 20 refer to entities in a close distance to the anaphor: the same or preceding tutor or student turn. The majority of long distance references are by means of nominal anaphora whose antecedents can be found two or more turns back in the dialogue. Around 25% of the references to terms have antecedents in the task description. The majority of references to proof steps (around 93%) were within the scope of the same or previous student turn. Only nominal references were used to refer to proof steps further in the preceding dialogue. Interesting to note is that the tutors did not request explicit clarifications of the scope of reference to proof steps, even if the scope encompassed a number of steps; much as the English ‘‘hence’’ or ‘‘thus’’, the German ‘‘somit’’ or ‘‘damit’’ can refer to a larger part of a constructed proof. This suggests that tutors cooperatively interpreted students’ contributions and tended to focus on the task progress, rather than on rigour or on closely monitoring the students’ mental representation of the solution.

As mentioned previously, the low overall number of referring expressions available for analysis does not allow us to draw definitive conclusions nor to develop a scalable reference resolution algorithm. However, preliminary observations based on the available data can be summarised as follows: Anaphoric references have for the most part a local scope. In most cases, the referent occurred in the same or preceding student or tutor turn with respect to the anaphor. The structure of mathematical expressions is a strong indicator in identifying antecedent search space; see also Section 3.2.1.2 (p. 95). This holds both in the case of noun phrase references to topographical substructures of mathematical expressions (‘‘inner parenthesis’’ or ‘‘left side’’) as well as in the case of quantified phrases (as in the ‘‘both complements’’ example).

Also relevant in antecedent search is the correctness status of the last student’s proof step. As a student develops a proof, the salience of propositions which form the proof (proof steps) changes. At the beginning of a dialogue, the most salient proposition is the goal formula in the task description. As the proof progresses, the most salient proposition globally is the last correct step and students tend to refer to this step. If the student makes several incorrect steps, no correct steps, and the tutor has not given away any steps, the goal formula in the task definition remains the most salient proposition even after several turns. The semantic content of the last tutor move also plays a role in reference resolution. If the last tutor’s turn contains a hint which gives away a step, the student is likely to continue from this step and so also to refer to it.

6.3.2 Modelling concepts relevant in reference resolution

The corpus analysis summarised in the previous section shows that two issues must be taken into account in designing a computational reference resolution algorithm for the proof tutoring domain: First, a comprehensive analysis of mathematical expressions is needed. Second, processing indirect referring expressions whose antecedents are elements of the symbolic language (terms or formulas or parts thereof) and which use typographical properties of mathematical expressions (‘‘left side’’), objects and relations that build up the expressions (‘‘both complements’’), and the expressions' structure signalled by grouping symbols (‘‘inner bracket’’), requires extensions to the domain interpretation process. Namely, the entities identified through mathematical expression analysis need to be included in the domain model. The extensions to the processing architecture are briefly outlined below.

Extensions to mathematical expression parsing In order to support resolution of references to (parts of) mathematical expressions, our mathematical expression parser is implemented in such way that it is capable of identifying all the relevant substructures of mathematical expressions. It parses the linear notation of mathematical expressions in the input into an expression tree of the form shown in Figure 3.2b (p. 98). The parser has access to knowledge on the type of arguments and results of operations in the relevant areas of mathematics. In our case, this is, for instance, the information that the subset relation (denoted by a specific symbol) takes two sets as arguments and the type of the result is a proposition or that the union operation takes sets as arguments and its result is an object-denoting type. Each node of the expression tree is marked (‘‘annotated’’) as to whether it denotes an operator or a variable; operator nodes are marked with the type of their result. The root node of the tree is marked with the information on the type of the entire expression (TERM, FORMULA, etc.). The expression tree enriched in this way is an input structure to subroutines relevant for reference resolution.

At the time of parsing we create a discourse referent for the entire expression, but not for every substructure entity relevant for anaphor resolution. Instead, the mathematical expression parser includes subroutines which *on demand* recover substructures of mathematical expressions in specific `part-of` relations with respect to the original expression as well as their types. Recall that these are also represented in our domain model; see p. 220 and the following section. The choice of substructures was motivated by systematic reference in natural language to mathematical expression parts (see Sections 3.2.1.2, p. 95, and 3.2.2.5, p. 132) and includes: (i) topographical features (such as ‘‘sides’’ of terms and formula), (ii) linear orders (‘‘first’’, ‘‘second’’ argument), (iii) structural groupings (bracketed subexpressions) with information on the level of their embedding. Execution of these subroutines is triggered by rules in the course of lexical semantic interpretation of the utterances; for instance,

Table 6.13: Examples of reinterpretation rules for indirect reference

Concept	Reinterpretation
`Side`	`Term` at side
`Bracket`	`Bracketed term`
`Operator`	`Term` headed by `Operator`
`Object`	`Term` headed by `Operator` of type `Object`
`Property`	`Object` with property

the meaning of ‘‘side’’ together with its modifier ‘‘left’’ in the semantic representation of the noun phrase ‘‘the left side’’.

Domain modelling As illustrated in Section 3.2.2.5 (p. 135) and earlier in this section, informal mathematical language admits of referring to elements of mathematical notation using expressions of a metonymic flavour. By saying ‘‘the left side’’ of a formula, we do not mean literally the side, but rather the term on the given side of the main operator in the expression. The use of such metonymic expressions is so systematic in mathematics when referring to mathematical notation and they are such an integral part of the mathematical terminology that it is justified to think of them as quasi-synonyms of the concepts evoked by the entities to which they refer. Thus, in line with this observation, we encode *metonymy rules* as part of the domain model. The rules enable interpretation of utterances with certain sortal restriction violations by encoding reinterpretions of concepts evoked by metonymic words. This approach is analogous to the rule-based approach to metonymy proposed by Fass (1988), except that here rules are strongly specific to the domain of mathematics.

Table 6.13 shows examples of the reinterpretation rules encoded based on the phenomena found in our two corpora. The first rule means that the concept `Side` (left or right) may be alternatively interpreted as referring to a left or right term, respectively, of an expression in the previous discourse (as in ‘‘the left side is equal to...’’). The topographical properties of mathematical expressions are encoded as features of nodes of the parsed mathematical expressions (discussed earlier); thus an expression with the given property can be found by analysing mathematical expression parse trees. The second rule means that `Bracket` can be interpreted to refer to a term enclosed in brackets (as in ‘‘the inner parenthesis is equal to...’’); again presence or absence of bracketing is marked as a feature of mathematical expression tree nodes. The next two rules mean that an `Operator` can be interpreted as a term headed by the given operator (as in ‘‘for the complement we have...’’) and that an `Object` type can be interpreted as a term headed by an operator which builds an object of the given type. The last rule means that a property can be interpreted as the object which has a given property (as in ‘‘for the left side it holds that...’’). Multiple rules can be applied in the course of reinterpretation until a concept of a matching type is found. For example, the nominal reference ‘‘diese(r) Menge’’ (*this set*) referring to the expression $(S \cup R) \circ S^{-1}$ in the example (100) earlier in this section (p. 237), can be resolved by applying the rule ‘‘`Term` headed by `Operator` of type `Object` for `Object`’’.

6.4 Cooperative correction of mathematical expressions

In Section 3.2.1.5 (p. 106) we showed examples of flawed mathematical expressions constructed by the students (Table 3.3). We categorised the errors (Table 3.2) and identified their possible sources (Table 3.3). In principle, in a dialogue environment, clarification subdialogues could be initiated to point out imprecise wording or errors, and to elicit clarification or correction. Clarification subdialogues may, however, turn unwieldy making the dialogue tedious. This would be particularly undesirable when the problem solving skills of the student are otherwise satisfactory. A better solution would be to attempt to cooperatively correct what appears to be an error, or to resolve ambiguity, while allowing the student to concentrate on the problem solving task.

Using domain knowledge and reasoning, proof contributions may be evaluated for correctness. However, finding the intended reading of erroneous or ambiguous statements and the decision as to whether the flawed statement should be corrected by the student is pragmatically influenced by factors such as the student's knowledge of the domain concepts and their prior correct use, correct use of the domain terminology or contextual preference for one reading over the others. On the one hand, in a tutoring context, it is important to recognise the student's intention and knowledge correctly. On the other hand, however, it is also important not to distract the student by focusing on all low-level errors. In the most ‘‘accommodating’’ approach, erroneous and ambiguous expressions evaluated as correct in one of the readings could be accepted without requiring clarification on the part of the student, thus making the dialogue progression smooth and maintaining focus on problem solving. As we already pointed out earlier, the tutors did not tend to focus on low-level errors and accepted proof contributions even with flawed notation.

In order to facilitate this kind of cooperativity, we developed a strategy for flexible mathematical expression analysis and correction. When a malformed expression is encountered, we attempt to identify and correct type errors and logical correctness errors. The goal in this approach is to delay clarification, while making sure that the student's intentions remain tractable. The ultimate decision whether to accept an erroneous or ambiguous utterance (a strategy suitable for competent students) or whether to issue a clarification request for the student to disambiguate the utterance explicitly, is left to the tutoring component (recall the overall architecture presented in Section 1.2, p. 35).

The correction strategy we tested is based on introducing informed modifications to erroneous expressions with the goal of finding the plausibly intended correct form. The highest-ranked well-formed hypothesis generated by the algorithm is assumed to be the intended expression and is interpreted in the problem-solving context, so that its correctness and relevance can be addressed, while the fact that the expression was malformed can be merely signalled to the student by pointing at the error. Finding meaningful modifications of a malformed expression is guided by the expression's error category. With each error category shown in Table 3.2 (p. 108) we associate a set of replacement rules and apply them to a malformed expression with the goal of improving its category. That is, from a syntactically ill-formed expression we try to obtain a syntactically well-formed one and from an expression with a type mismatch we try to obtain a well-typed expression. The selection of replacement rules is motivated by an analysis of possible sources of errors in the erroneous expressions in our two corpora; see Table 3.3 (p. 110). The correction algorithm and a pilot evaluation are outlined in the following sections.

Correction algorithm

The correction algorithm assumes that mathematical expressions are parsed by a tree-building algorithm; for experiments we used the same parser as the one we use throughout this work (see Section 5.2.2.3 (p. 185) and the extensions outlined in Section 6.3.2 earlier in this chapter). For unbracketed operators of the same precedence, all possible bracketings are considered (for instance, the expression $A \cup C \cap B$ is ambiguous between $(A \cup C) \cap B$ and $A \cup (C \cap B)$). For every tree node, the parser stores information on whether the subtree it heads is bracketed in the original string and whether the types of arguments are consistent with the expected types. The output of the parser is a formula tree with nodes marked as to type compatibility and bracketing where applicable.

Erroneous expressions are systematically modified by applying operators considered suitable for removing the reported error. The resulting new expressions are then categorised by consulting the formula analyser and, if needed, a reasoner to check the new expression's correctness. Since the latter may be an expensive step, the generated hypotheses (candidate corrected expressions) are ranked and tested in the rank order. The process can be terminated at an intermediate stage if calls to the reasoner are becoming too costly. The process can also continue iteratively if needed, resources permitting.

The hypotheses are ranked using three ordered criteria: (1) the error category of the modified expression, (2) the number of operators applied so far, and (3) the structural similarity of the hypothesis to the expressions in the previous context. Similarity is approximated by counting the instances of common operators and variables. The context consists of the goal expression, the previous proof step, and possible follow-up steps generated by the reasoner.

The pseudo-code of the algorithm is shown in Figure 6.5. The algorithm takes two arguments: the original *Expression* (parsed by a mathematical expression parser) and a set of expressions representing *Context*. An expression can be of one of four categories: Category 1 is a logical error (an expression is well-formed and well-typed, however, a weaker or stronger statement is expected), Category 2 is a semantic error (an ill-typed expression), Category 3 is an ill-formed expression, and Category 0 is a valid correct expression. The procedure consists of three parts. In the first step, for ill-typed expressions, operators associated with the error category are selected. In the second step, replacement operators – see Table 3.3 (p. 110) – are applied to the original formula, possibly at multiple places. The application of operators addressing ill-typed expressions is limited to those places where the parser reported a type error. New expressions resulting from each replacement are collected in *Hypotheses*, excluding results considered *Trivial* (for instance, an equation with identical left and right sides or applications of idempotent operators to identical arguments), and their error category is returned by the parser (Parse). In the third step, the hypotheses are assessed in a two-pass evaluation. First, similarity to the expressions in *Context* is computed. For expressions which were originally false statements, a call to the reasoner is made. Since the latter can be expensive, the expressions obtained by applying operators are ordered according to contextual similarity, prior to invoking the reasoner. The evaluation of the ordered list of expressions can be stopped any time if resources are exhausted; this criterion is denoted by the condition *Limit*. The procedure terminates when the problem is solved, that is, the category of some modified expression is improved, when no more operators can be applied, or when resources are exceeded. If one of these cases holds, the ordered list of *Hypotheses* is returned; otherwise, application of operators to the newly created expressions is repeated. Several limits on resources involved can be considered, including: (i) maximum number of modified formulas created, (ii) a time limit (checking correctness of an expression can be time consuming), (iii) number of calls to the reasoner, (iv) a limit on the number of errors addressed (or operators to be applied).

```
Data: ME (original expression), Context (expressions in prior discourse)
// Collect operators
switch ME Category do
    case 3
        Hypotheses ← List(Results of ME analysis);
        Evaluate(Hypotheses, Context);
    end case
    case 2
        Operators ← Operators$_2$;
        Hypotheses ← Result of ME analysis
    end case
    case 1
        Operators ← Operators$_1$;
        Hypotheses ← Result of ME analysis
    end case
endsw
Iterate: // Apply operators to the original ME
foreach (Hypotheses,Operators) do
    New-formulas ← Apply Operator to Hypothesis;
    foreach New-formula in New-formulas do
        if not Trivial(New-formula) then
            New-formula-parse ← Parse(New-formula);
            Hypotheses ← Hypotheses ∪ New-formula-parse;
        end if
    end foreach
end foreach
Evaluate: // Decide if continue required/affordable
Evaluate Hypotheses within Context;
Sort Hypotheses by evaluation score;
foreach Hypotheses do
    while not Limit do
        if Hypothesis Category==1 then
            if Hypothesis correct then
                Hypothesis Category ← 0
            end if
        end if
    end while
end foreach
Sort Hypotheses by evaluation score;
if not ME Category == 3 and not Best hypothesis category superior to original
and not Limit and New expressions built then
    goto Iterate
end if
return Hypotheses
```

Figure 6.5: Pseudo-code of the mathematical expression correction algorithm

Preliminary evaluation

Below we present a preliminary evaluation of the proposed correction algorithm. The algorithm was tested on a sample of ill-typed and false expressions from the corpora and on a larger set of expressions into which errors of the above-mentioned categories were introduced in a controlled way.

Evaluation data The evaluation data we used stemmed from two sources: a set of recurring erroneous expressions from the corpora (*Corpus*) and a set of expressions obtained by systematically introducing errors to valid expressions, according to our categories (*Constructed errors*). The *Corpus* data set contained 8 most representative cases of the kinds of errors that occurred in the data. Multiple occurrences of similar expressions were not included; by ‘‘similar’’ we mean expressions of the same structure which differ only by identifiers. *Constructed errors* were created in the following way: First, from the corpus we extracted valid formulas which occurred in proof contributions evaluated by the tutor as correct; there were 71 unique expressions. Then, for each of these we generated a set of erroneous expressions by systematically changing the operators and identifiers according to error categories. For practical reasons, we introduced at most two errors into one expression in order to make the correction task manageable. For example, for the valid expression $A \cap B \subseteq P(A \cap B)$ we generate, among others, the following erroneous expressions:

Dual operator errors	$A{\cup}B \subseteq P(A{\cap}B)$	
	$A{\cap}B \subseteq P(A{\cup}B)$	
Confused operators	$A{\cap}B \in P(A{\cap}B)$	
	$A{\cap}B \subseteq K(A{\cap}B)$	
	$A{\cap}B \subseteq P(A{\cap}P)$	(two errors)
Confused identifiers	$A{\cap}P \subseteq B(A{\cap}B)$	
	$A{\cup}P \subseteq P(A{\cap}B)$	(two errors)
	$X{\cap}B \subseteq P(A{\cap}B)$	(X: arbitrary identifier not in context; simulates typos)

From the generated set of erroneous expressions, we built the *Constructed errors* data set for evaluation by randomly selecting 100 expressions in which the number of operators was between 3 and 10.

The choice of the two data sets was motivated by complementary factors: The *Corpus* sample is intended to give an insight into the algorithm’s effectiveness when applied to authentic errors. This sample is however very small, 8 instances. The *Constructed errors* sample is intended to assess the prospect for the algorithm based on a larger set of errors of the same type.

Table 6.14: Results of formula correction

Evaluation data set	Unique result	Ambiguous	Target in top 10
Corpus	2	6	6
Constructed errors	0	100	64

Table 6.15: Results of hypothesis generation for *Constructed errors* data

Evaluation measure	Min	Max	Mode
Number of hypotheses generated	5	38	18
Position of target expression in hypotheses list	1	18	14

Limits applied In order to carry out formula modifications within feasible resources, we applied two limits: (i) to keep the set of generated hypotheses manageable, the number of considered errors was restricted to at most two in one formula (this level of complexity accounts for most of the errors that occur in the corpus), (ii) the calls to the reasoner were limited to five since this is the most expensive part of the algorithm; we prefer this qualitative criterion over a time limit criterion because the results are not influenced by the implementation of the reasoner. A component of Scunak (Brown, 2006b) was used in this pilot experiment. Outputs were verified manually.[13]

Results The results are summarised in two tables. Table 6.14 shows the overall performance in terms of the number of corrected expressions for which a single correct hypothesis was found (Unique), those for which multiple hypotheses were found (Ambiguous), and the number of cases where the target expression was among the top 10 ranked candidates. Table 6.15 shows two results for the larger evaluation set: a measure of effort required to generate corrections in terms of the number of generated hypotheses and the position of the intended formula in hypotheses list. Note that the top position in the list does not imply that a unique solution is found since multiple candidates may obtain the same final rank.

Discussion The results show that automating formula correction is a non-trivial task. For an objective sample of complex expressions with errors (three to ten operators, up to two errors per expression) the algorithm was able to place

[13] The author of this thesis verified half of the data set. The other half was verified by the collaborator in this work Dr Helmut Horacek.

the intended expression in the top ten hypotheses in 64% of the cases. However, there is no guarantee that further evaluation of the top candidates by a reasoner yields a unique candidate. The two unambiguously corrected expressions from the *Corpus* sample (see Table 6.14) were very simple and only one change of an incorrect operator was applicable. The results on the *Constructed errors* data set show that both the hypothesis generation (large range of generated hypotheses) and the ranking (most targets below top-10 ranked hypotheses) needs an improvement. Error analysis suggests that three factors could lead to improvements: exploiting the reasoner further (for instance, by querying for further formulas entailed by the formulas in context; this would of course require a reasoner with proof automation), adding more contextual information (for instance, analysing the kinds of errors which a learner previously made), and improving the similarity calculation (incorporating information on structural similarity, rather than just identifier overlap).

6.5 Summary

As we have shown in Chapters 3 and 4 and contrary to expectation, students' mathematical language is rich in interesting phenomena and diverse in terms of patterns of verbalisation. Only a subset of all the linguistic phenomena can be addressed within a scope of one thesis. In order to show the general feasibility of provisioning language processing capabilities for a tutorial dialogue system for proofs, in this chapter we opted for the breadth of coverage, addressing a wide range of phenomena, rather than focusing on a narrowly defined linguistic problem and modelling it in depth. For the same reason, we chose to address subsets of phenomena at different levels of computational analysis: syntactic, semantic, and discourse, guided by two criteria: frequency of occurrence in the corpora and complexity of computational modelling.

Among the basic phenomena which need to be modelled and which frequently recur in our corpora are those related to the syntactic properties of the input language and its peculiarities due to the mathematical domain. We have shown how we model basic German syntax in combinatory categorial grammar and gave a categorial account of informal mathematical language with embedded formal notation, including its idiosyncratic domain-specific language constructions. At the semantic level we focused on linguistic imprecision and ambiguities in interpretation which it entails. We have shown how a lexical resource, a semantic lexicon, can be exploited to link imprecise concepts with domain concepts via a linguistically-motivated domain ontology. The stepwise interpretation process is well-motivated in that it reflects the observations on how mathematical objects are conceptualised in the course of learning (see Section 3.2.2.4, p. 126). Among complex discourse phenomena, we model a contextual operator, ‘‘the other way round’’, which frequently occurs in spontaneous speech and which has been also found in our corpus data. Both the semantic lexicon and the transformations employed in ‘‘the other way round’’ reconstruction exploit the dependency structures which we use to represent natural language semantics. This supports our choice of tectogrammatical representation of meaning, proposed in Chapter 5, as an appropriate level of abstraction for modelling a wide range of semantic phenomena. Also at the discourse level, we analyse reference phenomena and show how to extend our domain model to account for indirect reference specific to mathematical discourse. Finally, we test our observations on common errors in mathematical expressions (outlined in Section 3.2.1.5, p. 106) in a preliminary error correction method whose purpose is to support cooperative interpretation.

Our approach in this chapter has been mainly qualitatively oriented and served the objective of showing feasibility of computational interpretation by the range of phenomena addressed. We showed implemented proof of concept models or performed corpus-based studies as preliminary step towards computational implementation. Evaluations have been of small-scale, pilot character. As is clear from this chapter, the semantic interpretation methods we propose depend mainly on hand-crafted resources (grammars, lexica, ontologies, rules) and the methods employed are deterministic in nature. Crucial is, however, that input can be parsed. In order to gain insight into the prospects for larger-scale computational interpretation, in the next chapter we perform a quantitative evaluation of the parser component, the element of the architecture on whose output semantic interpretation relies.

Chapter 7

Prospects for automated proof tutoring in natural language

This chapter reports on evaluation experiments designed with the goal of assessing the potential of deep processing resources for computational understanding of students' mathematical language and drawing conclusions about the prospects for natural language interaction in German dialogue-based proof tutoring systems. We focus on the coverage of the parsing component which is the key part of the proposed input interpretation architecture (Chapter 5). Existing corpora of learner proofs (Chapter 2) are used as data for an intrinsic evaluation of the parser's performance. Before presenting the results, we motivate the choice of the evaluation methodology, the scope of the evaluation, and the design of the experiments.

7.1 Methodology and the scope of evaluation

Holistic approaches to evaluating tutoring systems use empirical methods – laboratory or field experiments – to show a relationship between an intervention involving computer-based instruction and the students' outcomes (Mark and Greer, 1993; Self, 1993; Baker and O'Neil, 1994). The Stanford tutoring systems, including the proof tutoring environments, have been evaluated in this way since the 60s; see, for instance, (Suppes and Morningstar, 1972; Suppes, 1981). Such ''end-to-end'' evaluations presuppose, of course, that a complete implemented system exists and, crucially, that it is robust enough to handle new data in a live study. If a complete system is not available, partial Wizard-of-Oz experiments (see Section 2.2, p. 64) may serve as a setting for evaluating parts of a larger system while emulating unimplemented components.

The project of which this thesis was part focused on *basic* research questions in modern technology for dialogue-based tutoring of mathematical proofs rather than aiming at a deployable system. Several *proof-of-concept* studies have been conducted within the project in order to assess the validity of the proposed methods on component-by-component basis: tutoring strategies (Tsovaltzi, 2010), fragments of the dialogue model (Buckley, 2010), granularity judgement models (Schiller et al., 2008), and recently also proof representation and reasoning (Autexier et al., 2012). Integrating the proof-of-concept modules into a working experimental system would be an interesting task in itself, but it is outside of the scope of this thesis. At the present state, even in a partial Wizard-of-Oz simulation most of the anticipated system's functionality would have to be taken over by a human facilitator, making the experiment logistically complex and costly. Therefore, instead, in this work we follow the same method of component-based evaluation and use *intrinsic criteria* to evaluate deep-parsing German CCG fragments based on corpora.

Intrinsic evaluation (Galliers and Spärck Jones, 1993) focuses a component's objective, rather than its role in a larger setup (extrinsic).[1] Precision and recall are often used as measures in intrinsic parser evaluation; see, for instance, (Grishman et al., 1992; Mollá and Hutchinson, 2003; Carroll et al., 2003). An evaluation which is closest to ours in terms of the application domain has been performed by Dzikovska et al. (2005). The authors report 62% coverage and 68% precision results for syntactic and semantic parsing of the LeActiveMath corpus of English tutorial dialogues on differentiation (Callaway et al., 2006).[2] The results were obtained by manually extending the lexical base of the TRIPS grammar (Allen, 1995), a wide-coverage parsing resource for dialogue, to support the LeActiveMath data. Similarly to this work, we use Wizard-of-Oz corpora (Chapter 2) to investigate the growth of parsing coverage with an increasing size of grammar resources as well as the amount of parse ambiguity generated by the grammars.

Note that in a stepwise deep processing approach based on manually constructed lexicalised resources and without robustness features, parsing is the critical part of the input interpretation component: If the parser fails, domain-specific interpretation, the next step of the processing pipeline (Chapter 5), cannot proceed. Once a parse is found, assigning a domain-specific reading is a deterministic (rule-based) process. Grammar coverage is thus critical to the usability of a system based on deep semantic processing. Therefore, in order to assess the outlook for deep processing-based interpretation, we focus on the performance of the manually constructed parsing grammars.

[1] For an overview of parser evaluation methodologies see also, for instance, (Carroll et al., 1998).

[2] Based on the reported results it is not clear whether utterances or turns (possibly multi-utterance) were parsed and what proportion of the parsed units were unique verbalisations.

Table 7.1: Summary of the utterance types distribution

Utterance type	C-I&C-II
	Unique / Total
Solution-contributing	465 / 735
Proof contribution	450 / 719
Proof step	407 / 640
Logic and proof step components	175 / 366
Domain & context	126 / 256
Meta-level description	16 / 186
Proof strategy	29 / 34
Proof status	7 / 29
Proof structure	7 / 16
Meta-level	15 / 16
Self-evaluation	7 / 7
Restart	4 / 5
Give up	4 / 4
Other	231 / 331
Help request	149 / 170
Yes/No	1 / 42
Cognitive state	30 / 31
Politeness/Emotion/Attitude	14 / 24
Discourse marker	1 / 22
Answer	19 / 20
OK	1 / 7
Address	6 / 6
Session	4 / 4
Agree	3 / 3
Self talk	2 / 2

The experiments we conduct are restricted to two types of *Proof contribution* categories. The reason for this is two-fold: First, it is the proof-contributing utterances that need a domain interpretation readable by a reasoning component; the interpretation strategy and the language processing methods proposed in Chapters 5 and 6 concentrate on this type of utterances. Second, the data in the remaining classes is sparse. Recall that in Chapter 4 we classified the learner utterances into two broad types: *Solution-contributing* and *Other* (non-solution-contributing). The utterance types frequency distribution is summarised in Table 7.1.[3] If we exclude subcategories of *Other* which can be identified by a lexical lookup (*Yes/No*, *OK*, and *Discourse marker*) we are left with 8 subtypes of which only four have a frequency above 5% within their superclass (*Answer*, *Politeness/Emotion/Attitude*, *Cognitive state*, and *Help request*). The set of help requests could be considered for experiments, although, admittedly,

[3] For the full classification see Table 4.6, p. 165.

170 instances might not be a representative sample. While help requests could be also parsed using deep grammars, it is evident that this category is linguistically diverse, with mainly idiosyncratic verbalisations (type–token ratio of 0.88). Thus, grammar-based parsing might not scale. Moreover, since help requests are not passed to a reasoning engine for evaluation, but can be processed by the dialogue model directly, an alternative strategy worth exploring would be machine-learning-based classification.[4] The *Solution-contributing* class is likewise skewed. Only the *Proof step* category constitutes more than 5% of the class. Verbalisations of the remaining categories can be hardly considered representative (the *Meta-level* classes have between 4 and 7 instances and the remaining *Proof contributions* between 16 and 34 instances). Therefore, the evaluation we conduct encompasses only proof-contributing utterances, more specifically, *Proof contributions* of type *Logic & proof step components* and *Domain & context* as defined in Section 4.3.4.[5]

7.2 Design

We attempt to answer the following questions: First, beyond principled compositional semantic construction, is there advantage to deep processing students' input over parsing using resources which are easier to author?[6] Second, do the resources scale, that is, what can we tell about the prospects for natural language as input to proof tutoring systems based on processing the available data? To this end, we set up an experiment to analyse two aspects of parser performance: *parsing coverage* (proportion of parsed utterances from a test set) and *parse ambiguity* (number of parses found for a parsed utterance). The experiment consists of two parts: First, we analyse the growth of coverage in a pseudo cross-validation experiment on ''seen'' data (data used for grammar development). Second, we evaluate the performance of the same grammar resources on ''unseen'' data (not used for grammar development, a blind set).

It is clear that verbalisations of proof steps are linguistically diverse (type–token ratio of 0.49; see Table 7.1) and many verbalisations occur only once (48% in the *Logic & proof step components* class, 84% in the *Domain & context* class; see Figure 4.4, p. 171). Of course, considering that we build grammars by hand, we could model all the proof step utterances one by one or

[4] If the taxonomy we proposed in (Wolska and Buckley, 2008) were used, this would be a 7-way classification task.

[5] *Meta-level descriptions* are not included for the same reason: at 18 instances the sample is too small. When we refer to ''proof steps'' further in this chapter we mean the *Logic & proof step components* and *Domain & context* types.

[6] Arguably, writing regular or context-free grammars is less involved than writing resources in richer, more expressive grammar formalisms, such as HPSG, LFG, or CCG.

focus on specific linguistic phenomena of the German language.[7] Instead, for this evaluation, we select utterances to model based on a shallow quantitative measure: we do not model proof step verbalisations which are entirely idiosyncratic, but use only those verbalisations which, upon preprocessing, occur *at least twice* in the data up to the given point in the experiment. We will refer to these subsets of the data as ''modelled utterances'' or a ''development set''. At each cross-validation step, grammars are built based on modelled utterances stemming from an increasing number of dialogues (1 dialogue, 2, 3, and so on).

The motivation behind this setup is to simulate a partial Wizard-of-Oz experiment in which the parsing component is replaced by a human if it fails. In the envisaged scenario, we would systematically augment the grammar resources after each experiment session based on the data from the subject who just completed the experiment, a plausible approach. Since grammar development is a time-consuming task, for efficiency reasons a plausible pragmatic decision in such a setting would be to prioritise modelling those verbalisations which are observed to reappear – suggesting thereby to be *relatively more representative of the language* – with the view to gradually reducing the degree of wizard's intervention. For instance, one could decide to model utterances which appeared at least, say, five times in the data collected so far. Given the heavily skewed distribution of the proof step types (see Figure 4.4, p. 171), in the simulated experiment we set the frequency threshold at two occurrences for otherwise the development sets would be too small.

In the pseudo cross-validation setup, we parse *all* the utterances from the modelled set (seen data) using grammars constructed based on the modelled utterances.[8] Notice that unlike in proper cross-validation, in which data is partitioned into *disjoint* development and validation sets, here the evaluation sets constructed from the modelled utterances contain both utterances unseen at the given iteration (modelled, but not used to built the grammar at the given step) as well as seen items (items based on which the evaluated grammars have been built). The purpose of the evaluation on the modelled sets is to observe the *rate of convergence* to ceiling results (total number of modelled utterances) based on data that has been exhaustively encoded in a principled way (all the utterances from the seen evaluation sets parse into the expected representations). Next, we use the remaining proof step utterances, the single-occurrence verbalisations (unseen data), to observe the *generalisation potential* of the grammar. Analogous incremental evaluation is performed. The second part of the experiment is thus a proper blind evaluation. In the next section, the development data, the grammars, and the test sets are presented in more detail.

[7] We have shown how we model selected relevant phenomena of German in CCG in Section 6.1.

[8] Descriptive information on the development and evaluation sets follows in Section 7.3.

Table 7.2: Descriptive information on the grammar development set

	C-I	C-II	C-I&C-II
Number of dialogues in the development set	15	27	42
Number of unique utterances	21	56	61
Number of words	80	266	284
Number of unique types	24	54	57

7.3 Data

Out of the 57 dialogues 42 contain proof steps which overall occurred more than once. They comprise 622 proof step instances, among which, after preprocessing, there are 391 unique verbalisation patterns. 319 of these occurred once, leaving 72 utterance patterns for developing the evaluation grammar. 10 clearly ungrammatical utterances were excluded.[9] The pattern consisting of a single noun phrase denoting a domain term was also excluded.[10] The remaining 61 utterances from 42 dialogues were used as the development set.

7.3.1 Preprocessing

Utterances in the development set were preprocessed as described in Chapter 5. Domain terms and mathematical expressions have been identified and substituted with symbolic tokens. In the case of mathematical expressions, the tokens represent the expression's type (TERM, FORMULA, _FORMULA, etc.), in the case of domain terms, they include grammatical information about case, number, gender, and the type of article (definite/indefinite/none), for instance, domainterm:def-sg-f-dat for a definite, singular, feminine, Dative noun phrase. Two multi-word units, ‘‘genau dann wenn’’ (*if and only if*) and ‘‘so dass’’ (*such that*) have been represented as single tokens. Table 7.2 summarises descriptive information about the development set.[11]

Figure 7.1 shows the distribution of utterance lengths (pattern lengths) in the modelled set. The majority of utterances from both development sets are between three and five tokens. The binary relations corpus contains a larger

[9] Examples of ungrammatical forms include: ‘‘dann gilt fuer die linke seite wenn formula’’ (main clause of the embedded sentence missing) or ‘‘term gilt demnach wenn formula und formula’’ (semantic type conflict between the subject ‘‘term’’ and the predicate‘‘hold’’). The grammar can parse the latter utterance once ‘‘demnach’’ is added as a lexeme to the prepositional adverbs category and if ‘‘term’’ is replaced with ‘‘formula’’.

[10] After preprocessing to a single token, DOMAINTERM (NP category) this is a trivial case.

[11] Note that here and further *type counts* rather than instance counts will be reported. Note also that whenever we use the word ‘‘utterance’’ further in this chapter, we really mean *utterance pattern*, an utterance preprocessed as described here. Both terms will be used interchangeably.

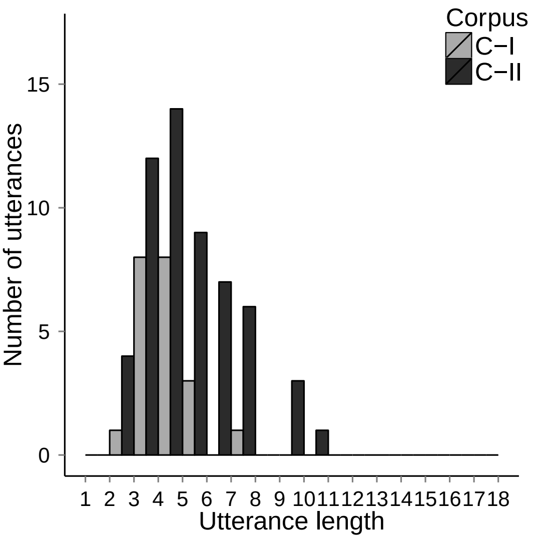


Figure 7.1: Histogram of the modelled utterance lengths (in tokens).

number of longer utterances than the naïve set theory corpus. Considering that this suggests a wider variety of linguistic phenomena in C-II, we expect that resources stemming from C-II data will provide better generalisation, thus better coverage, on unseen data than the resources stemming from C-I.

7.3.2 Evaluated grammars

Dialogue utterances from the modelled set have been exhaustively encoded in OpenCCG as follows: The set of atomic categories comprises the standard four types: S for sentence/clause types, NP for noun phrases, N for common nouns, and PP for prepositional phrases. A set of basic categories for mathematical expressions, noun phrases, common nouns, and articles has been encoded as a core *shared lexicon*. *Dialogue-specific lexica* of syntactic categories covering the phenomena found in the modelled utterances have been created for each dialogue in the development set. The shared lexicon, dialogue-specific lexica, and performance optimisation, are outlined in the sections that follow.

7.3.2.1 Shared lexicon

Four lexical families – mathematical expressions, noun phrases, common nouns, and articles – constitute the core set of categories available at *each* step of the iterative evaluation.

Mathematical expressions The grammar encodes three categories for truth-valued mathematical expressions: a sentence/clause type, S, and two NP\NP types for expressions of type _FORMULA: one with the ''such that'' reading, adding the formula's predication to the logical form via *GeneralRelation*, and the other adding a predicate, rather than a dependency relation, serving as the head of a dependency structure.

Mathematical object-denoting expressions, terms, obtain two categories: noun phrases and common nouns. The former models constructions such as ''... weil S eine leere Menge ist'' (... *because S is an empty set*), while the latter, constructions such as ''Es gibt ein x ...'' (*There is an x ...*) or ''Es gibt ein $x \in B$'' (*There is an $x \in B$*) in which a symbolic expression of type FORMULA is a part of a phrasal constituent with the preceding natural language material (here, part of a noun phrase).

Mathematical function and relation symbols embedded within natural language text obtain both clausal and nominal reading, the latter to account for constructions such as ''wegen Distributivitaet von $\circ$'' (*because of distributivity of* $\circ$). Partial expressions (such as ''$\in A$'') obtain appropriate functional categories (''$\in A$'', preprocessed to _FORMULA, is of type NP\NP).

Noun phrases The noun phrase group comprises three categories: two atomic, NP, denoting object types (contribute HLDS predicates) and expletive uses of singular third person neuter pronoun ''es'' (not represented in the logical form). The third noun phrase category, NP/NP, encodes appositive constructions and adds an *Apps* (appositive) relation to the logical form.

Common nouns A single atomic common noun category, N, models bare nouns and mathematical terms.

Articles Articles are modelled with the standard category NP/N.

7.3.2.2 Dialogue-specific lexica

Aside from the shared categories included in all grammars, dialogue-specific grammars encode *only* the categories required to cover the modelled utterances found in the given dialogue. An overview of our approach to modelling basic linguistic phenomena has been presented in Section 6.1 (p. 206). The same syntactic categories have been *consistently reused* across dialogue-specific lexica when the syntactic contexts allowed that, in order to ensure that *the same phenomena are modelled the same way across dialogues*, thus minimising spurious ambiguity due to alternative encoding in the grammar.

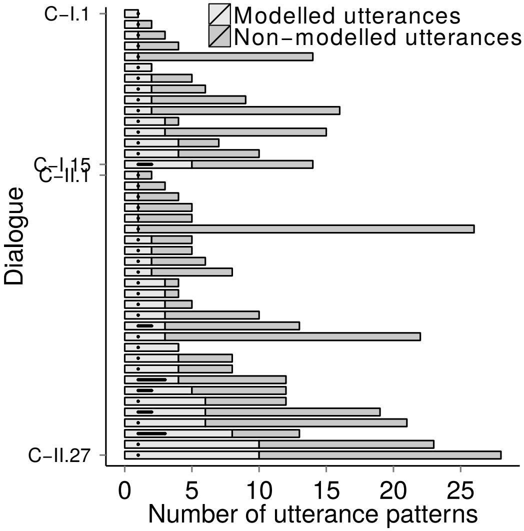


Figure 7.2: Distribution of modelled and non-modelled utterance patterns, sorted by corpus; horizontal lines denote the range of the number of parses in the modelled utterances per dialogue

7.3.2.3 Baseline

Performance of the CCGs is compared with performance of context-free grammars developed in an analogous setup. The CFGs were created using the NLTK toolkit (Loper and Bird, 2002) and parsed with the NLTK's Earley chart parser. The expectation is that the CCGs' lexicalised model provides better generalisations than the CFGs and, as a consequence, better coverage. However, the generalisation power is likely to come at a cost of ambiguity: we expect more ambiguous parses with CCG than with CFG.

7.3.2.4 Performance optimisation

Figure 7.2 shows the distribution of the modelled and non-modelled utterance patterns and the range of the number of parses per dialogue in the development set. Note that the x-axis shows the number of *distinct* utterances (pattern types) and not utterance instances (of which there were more than one instance in the case of all the patterns in the development set; see Section 7.2).

The performance of both CCG and CFG grammars was optimised on *per dialogue* basis: All the utterances were encoded in such way that, per dialogue,

Table 7.3: Verbalisations with multiple CCG parses in the development set

Utterance pattern	No. of parses
also gilt FORMULA und FORMULA	2
dies aber heisst FORMULA und FORMULA	2
FORMULA genaudannwenn FORMULA und FORMULA	2
also gilt FORMULA und FORMULA	2
laut DOMAINTERM gilt dann auch FORMULA	2
da FORMULA gilt nach DOMAINTERM formula	3
also ist TERM in TERM oder TERM in TERM	3

the expected (semantic) representations are correctly produced by the parser and that *the number of parses for the reading intended in the given dialogue is maintained at minimum*. While most of the utterance patterns have been encoded in such way that they produce a single parse (see Figure 7.2) the grammars do produce valid alternative derivations of a few utterances in the development set. Multiple CCG (and MMCCG) derivations are produced if a lexeme can be instantiated with multiple syntactic categories or if alternative applications of combinatory rules are possible.[12] There is a larger number of ambiguous parses in the binary relations corpus (this data set contains longer and more complex utterances; see Figure 7.1).

Utterances which yield more than one parse are listed in Table 7.3. Multiple parses are generated by ambiguous coordination which can be interpreted as taking wide or narrow scope, by a combination of coordination scope and preposition attachment or adverbial modification (‘‘auch’’ (*also*), ‘‘nun’’ (*now*), etc.), or by structurally ambiguous clausal scope. The three readings of the utterance ‘‘da FORMULA gilt nach DOMAINTERM FORMULA’’ are: ((da FORMULA) (gilt nach DOMAINTERM FORMULA)), ((da FORMULA gilt) ((nach DOMAINTERM) (FORMULA))), and (((da FORMULA gilt) (nach DOMAINTERM)) (FORMULA)). The latter two are artefacts of constructions of type ‘‘FORMULA gilt nach DOMAINTERM’’ (*FORMULA holds by DOMAINTERM*) and ‘‘nach DOMAINTERM FORMULA’’ (*by DOMAINTERM FORMULA*) for which different preposition categories are needed. Plausible alternative parses, illustrated above, were preserved. Otherwise, derivations were controlled in a standard way through features and MMCCG’s modes on slashes. The full grammar covering the modelled utterances from both corpora consists of 65 distinct complex categories grouped into 19 lexical families (sets of categories of syntactically related lexemes).

[12] Alternative compositionally ambiguous parses may produce equivalent logical forms.

7.3.2.5 Grammar development sets used in evaluation

Dialogue-specific CCGs built for the modelled utterances from each of the 42 dialogues have been grouped into four evaluation resources:

1. C-I resources: model C-I dialogues,
2. C-II resources: model C-II dialogues,
3. C-I&C-II in the data collection order (C-I&C-II-*dco*): C-I resources added first, followed by C-II resources,
4. C-I&C-II in a random order (C-I&C-II-*ro*): C-I and C-II resources combined in randomised order.

Cases (1) and (2) simulate the situations in which, respectively, only C-I and C-II data were available. Cases (3) and (4) represent the settings in which both corpora are available, with case (3) corresponding to the chronological order of our data collection and, more importantly, the distinction between the two mathematical domains of the data collection experiments.

At each cross-validating iteration, grammars are augmented by adding resources needed for parsing all the modelled utterances from the dialogue included at the given iteration step. The added resources comprise entire *lexical families*, that is, all the syntactic categories for the lexemes occurring in a given modelled utterance. A more conservative approach would be to include only the one category which models the specific syntactic context appearing in the given utterance. This, however, would result in grammars over-tuned for the specific utterances added to the evaluation at a given step and would not give an insight into the generalisation potential of the CCG grammars.

Considering the conclusions of the quantitative analysis in Chapter 4, which showed, at a shallow level, that the language in C-I and C-II differs strongly, we expect the grammar based on C-I and C-II combined in random order, that is mixing resources from the two corpora, to yield the best performance.

7.3.3 Test sets

Performance of the four grammars is tested on modelled and non-modelled *within-vocabulary* utterances grouped into ‘‘seen’’ and ‘‘unseen’’ sets:

1. C-I-seen, C-II-seen, and C-I&C-II-seen: comprise modelled utterances from C-I, C-II, and C-I and C-II combined, respectively,
2. C-I-unseen, C-II-unseen, and C-I&C-II-unseen: comprise non-modelled utterances from C-I, C-II, and C-I and C-II combined.

While the seen test sets do contain utterances based on which the grammar has been built, in the incremental setup, at each iteration only the lexical

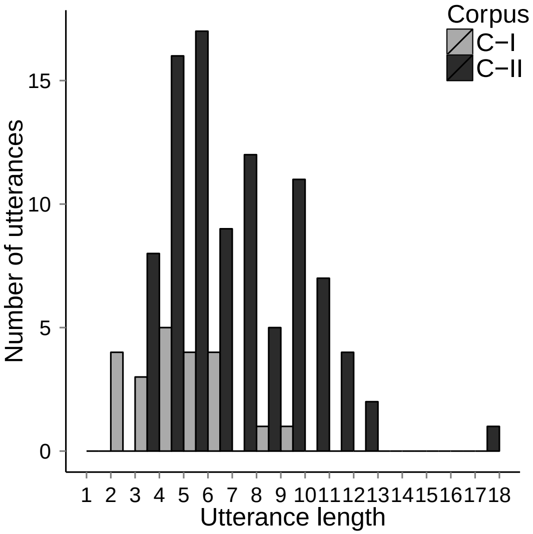


Figure 7.3: Histogram of the unseen utterance lengths (in tokens)

categories needed for *the given number of dialogues* are used. Thus, at each iteration of the ‘‘seen’’ evaluation, the grammar is tested on data from which the lexical categories stemmed *and* on the remaining data from the seen set which at the given iteration step is effectively unseen. Only at the final iteration step is the evaluation performed on the complete seen data set alone.

The unseen test sets consist of proof steps which occurred only once in all the 50 dialogues which do contain proof steps. 7 clearly ungrammatical utterances have been excluded. Only *within-vocabulary utterances*, *relative to the complete development sets*, have been included in the unseen test sets since parsing utterances with out-of-vocabulary (oov) words fails trivially.[13] The resulting unseen data set contains 114 utterances in total.

Figure 7.3 shows the utterance lengths in the blind sets. Not surprisingly, by comparison with the modelled set (see Figure 7.1), single-occurrence utterances are longer (more complex). We thus expect a significant drop in coverage by comparison with seen data. Table 7.4 summarises descriptive information on both test sets. 10 cross-validation rounds on different random permutations of the development dialogues are performed at each iteration step.

[13] In a basic deep-grammar parsing setup with no robustness measures, as performed here, oov words are not supported, that is, the parser fails. Note that in the incremental setup, evaluation on the incomplete seen sets will also cause parser failures due to oov words; parser failure rates due to oov words will be reported.

Table 7.4: Descriptive information on the test sets

	Seen data			Unseen data		
	C-I	C-II	C-I&C-II	C-I	C-II	C-I&C-II
Number of utterance patterns	21	56	61	22	92	114
Number of words	80	266	284	98	605	703
Number of types	24	54	57	26	48	49

7.4 Results

The results are summarised in four parts: First, we look at the coverage. Growth of coverage with an increasing number of dialogues is plotted per grammar resource. Variance of measurements obtained in the 10 cross-validation rounds is presented as box plots.[14] Subsets of numerical results – at 25%, 50%, 75%, and 100% of the data set – are statistically compared. The asymptotic Mann-Whitney-Wilcoxon U test, adjusted for ties, (α=0.05) was used due to a relatively small number of observations and because parametric assumptions were violated for most of the compared distributions. Parse failures due to vocabulary outside the lexicon (out-of-vocabulary error rates) are summarised. The analysis is performed for the seen data (Section 7.4.1) and the unseen data (Section 7.4.2). Next, parse ambiguity based on *full* grammars is plotted (Section 7.4.3). Finally, the overall performance of the CCG parser is summarised as percentage of test sets parsed and percentage of proof-contributing utterances parsed per dialogue (Section 7.4.4).

7.4.1 Coverage on seen data

Growth of coverage on seen data is shown in Figure 7.4 (p. 267). The rows show the evaluated resources and the columns the results for the three test sets. Ceiling values are marked with dashed horizontal lines: 21 for C-I-seen, 56 for C-II-seen, and 61 for C-I&C-II-seen data.

Two general trends can be observed based on these visualisations: First, on average, in all the cases the CCG grammars converge to ceiling values faster, that is, as expected, generalise better. Second, at around 50% of all the data sets, the performance of both grammars is characterised by substantial variance, that is, performance is strongly dependent on which dialogues are included in the data set. (Recall that the *dialogues*, not utterances, in the development sets have been sequenced randomly into 10 permutations.) This

[14] The same type of box plots are used throughout: hinges at Q1 and Q3, Tuckey whiskers (outliers outside 1.5*IQR), sample means marked with circles.

confirms the previous observation, formulated based on shallow analysis in Chapter 4, that the proof language is indeed diverse and differs from subject to subject; consequently, the individual subjects' data require different lexica or phrase-structure rules. As a result, the rate of convergence is also strongly dependent on the content of the development set. Moreover, the variance of the CCG results appears greater than that of the CFGs', which means that the convergence rate of the CCG parser is more unstable and more sensitive to changes in the data based on which the grammar is built.

Grammars based on C-I alone yield the poorest performance. C-I CCGs tested on seen C-II data do not reach the coverage of even 50% (the final C-I grammar parses 27 utterances on average out of 56). This is not surprising: the C-I resources are built based on only 21 utterance patterns (see Table 7.2) which are moreover shorter than the utterances found in C-II (see Figure 7.1).

Grammars based on C-II yield better performance. The complete C-II CCG misses only two utterances from C-I. Around 50% into the development set, C-II CCGs cover at least around 80% of all test sets. The combined resources reach at least around 70% coverage at 50% of the development sets. As expected, results based on resources combined in random order converge faster than those based on resources built incrementally in the data collection order. While the *dco* results exhibit a slow linear convergence trend for both CCG and CFG, convergence of the CCG results based on *ro* resources is clearly superlinear.

Numerical comparisons of parsing performance on seen data are shown in Table 7.5. Mean numbers of parsed utterances per test set are shown for subsets of the resources and for the complete development sets (standard deviations in parentheses). The n_d values indicate the actual number of *dialogues* in the development set. N are the ceiling values: the number of *utterances* in the given test set. Statistically significant differences are marked in bold.

In almost all cases the CCG parser statistically outperforms the CFG baseline. In fact, all the marked differences were significant at a more conservative significance level, 0.01, than the one used for comparisons. No statistical difference in the case of C-I&C-II-*dco* resources tested on C-I-seen test set is clear: 25% of the C-I&C-II-*dco* data set contains already 10 out of the 15 dialogues in C-I and the ceiling value is reached already 50% into the data set.

Table 7.6 shows the proportion of parse failures due to out-of-vocabulary words. Again, we see that the full C-I lexicon covers only around half of C-II and the combined test sets, that is, around 50% of C-II utterances contain vocabulary which is not in C-I. By contrast, with C-II and the resources combined in random order, *ro*, oov rates drop to at most 19% already when 50% of the data is available and to around 10% at 75% of the development data. The higher oov error rates, 25-28%, on C-I&C-II-*dco* resources are consistent with

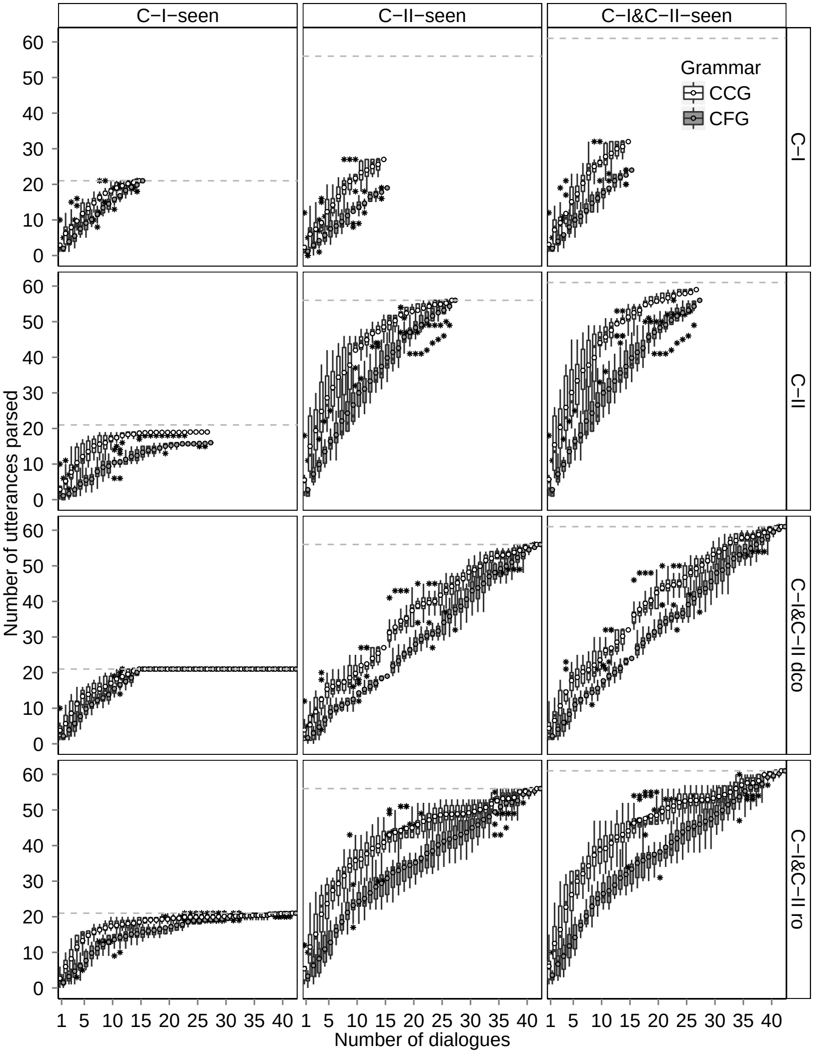


Figure 7.4: Growth of coverage on seen data.

Table 7.5: Mean coverage on the seen data in percentage of test set parsed

Grammar development set		C-I-seen (*N=21*)		C-II-seen (*N=56*)		C-I&C-II-seen (*N=61*)	
		CCG	CFG	CCG	CFG	CCG	CFG
C-I (n_d=*15*)	25%	**46.19**(13.31)	35.71 (9.82)	**16.43** (6.07)	10.18(4.30)	**19.34** (6.42)	13.11(4.15)
	50%	**70.48**(10.61)	55.24 (8.83)	**29.64** (5.88)	16.96(4.32)	**33.28** (6.09)	20.66(3.89)
	75%	**90.48** (7.68)	80.00 (7.00)	**41.61** (5.36)	26.25(2.77)	**45.57** (5.72)	30.98(2.88)
	100%	100.00 (0.00)	100.00 (0.00)	**48.21** (0.00)	33.93(0.00)	**52.46** (0.00)	39.34(0.00)
C-II (n_d=*27*)	25%	**71.90**(10.74)	37.62(11.75)	**61.43**(12.17)	39.46(9.66)	**59.51**(12.25)	36.23(8.87)
	50%	**87.14** (3.05)	57.62 (8.64)	**83.39** (3.39)	63.57(7.91)	**80.98** (3.30)	58.36(7.27)
	75%	**90.00** (1.43)	72.38 (3.56)	**94.46** (3.87)	86.25(5.93)	**91.48** (3.58)	79.18(5.44)
	100%	**90.48** (0.00)	76.19 (0.00)	100.00 (0.00)	100.00(0.00)	**96.72** (0.00)	91.80(0.00)
C-I&C-II *dco* (n_d=*42*)	25%	80.95 (9.78)	73.81 (9.10)	**35.32** (5.04)	25.54(3.10)	**38.62** (5.94)	29.02(3.81)
	50%	100.00 (0.00)	100.00 (0.00)	**69.25** (5.76)	52.32(3.49)	**71.77** (5.29)	56.23(3.20)
	75%	100.00 (0.00)	100.00 (0.00)	**87.30** (5.29)	78.21(7.04)	**88.34** (4.85)	80.00(6.47)
	100%	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00(0.00)	100.00 (0.00)	100.00(0.00)
C-I&C-II *ro* (n_d=*42*)	25%	**82.86** (9.33)	64.76 (9.33)	**63.93** (7.35)	43.39(6.87)	**63.61** (7.32)	43.44(5.92)
	50%	**93.33** (3.81)	83.81 (6.80)	**81.96** (5.78)	65.00(8.83)	**81.31** (5.35)	64.75(8.43)
	75%	96.19 (2.86)	94.76 (2.56)	**88.57** (3.68)	81.79(7.18)	**88.20** (3.72)	81.80(6.58)
	100%	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00(0.00)	100.00 (0.00)	100.00(0.00)

Table 7.6: Mean proportion of parse failures due to oov words on seen data

Grammar development set		C-I-seen (*N=21*)	C-II-seen (*N=56*)	C-I&C-II -seen (*N=61*)
C-I (n_d=*15*)	25%	0.53 (0.12)	0.82 (0.07)	0.79 (0.07)
	50%	0.35 (0.10)	0.68 (0.06)	0.65 (0.06)
	75%	0.14 (0.05)	0.58 (0.04)	0.54 (0.04)
	100%	0.00 (0.00)	0.50 (0.00)	0.46 (0.00)
C-II (n_d=*27*)	25%	0.34 (0.11)	0.39 (0.11)	0.40 (0.11)
	50%	0.18 (0.04)	0.16 (0.04)	0.19 (0.04)
	75%	0.12 (0.03)	0.05 (0.03)	0.09 (0.03)
	100%	0.10 (0.00)	0.00 (0.00)	0.03 (0.00)
C-I&C-II *dco* (n_d=*42*)	25%	0.19 (0.07)	0.61 (0.04)	0.58 (0.04)
	50%	0.00 (0.00)	0.28 (0.05)	0.25 (0.05)
	75%	0.00 (0.00)	0.10 (0.05)	0.09 (0.04)
	100%	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
C-I&C-II *ro* (n_d=*42*)	25%	0.17 (0.09)	0.33 (0.09)	0.33 (0.08)
	50%	0.07 (0.03)	0.15 (0.06)	0.16 (0.05)
	75%	0.04 (0.03)	0.09 (0.03)	0.10 (0.03)
	100%	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)

the corresponding C-I results: at 50% of the data only 7 C-II dialogues are included in the C-I&C-II-*dco* data set. The oov rates drop to around 10% at 75% of the *dco* set, the same level as with the *ro* set. The majority of parse failures on seen data are thus not due to differences in vocabulary, but due to different syntactic constructions in the development sets and test sets.

7.4.2 Coverage on unseen data

Growth of coverage on unseen data is shown in Figure 7.5. Ceiling values for the test sets marked with dashed horizontal lines are at 22 for C-I-unseen, 92 for C-II-unseen, and 114 for C-I&C-II-unseen. Performance of both the CCG and the CFG grammar is far from the ceiling values, however, the trends observed for the seen data are even more pronounced on unseen data.

In all the cases the CCG grammars' coverage grows faster. The CCG parser markedly outperforms the CFG parser on the C-II-unseen and C-I&C-II-unseen test sets. There is more variance in the performance of the CCG parser than of the CFG parser on the unseen C-II data and on the combined set, that is, again, the performance, and thus the rate of convergence of the CCG results, is strongly influenced by the content of the data set, again pointing at the diversity of linguistic phenomena. As with the seen data, grammars based on C-I data alone yield the poorest performance. There is little difference in performance between C-II and C-I&C-II grammars, which means that the C-I resources do not contribute much to the performance on unseen data. This again shows that the language in C-I is substantially different from the language in C-II.

Numerical comparisons of parsing performance on unseen data is shown in Table 7.7 (p. 272). The CCG parser consistently statistically outperforms the CFG parser, this time also on the test set based on C-I data. Both the CCG and the CFG parser performance is more stable on unseen data, however, the tendency towards more variance (less stability) in the performance of the CCG parser than of the CFG parser can be observed on unseen data as well.

Table 7.8 (p. 272) shows out-of-vocabulary parse failure rates on unseen data. With the complete C-I lexicon almost half of the parse failures on the unseen data from the same corpus and the majority of failures on the C-II unseen data and the combined set are due to oov words. However, much like in the case of the seen data, C-II grammars and the grammars combined in a random order yield only 10 to 30% oov failures given at least 50% of the resources. The majority of failures are thus due to syntactic constructions in the test sets which are not accounted for by the development data. With the complete C-II resources, all parse failures are due to this. The high oov error rates based on C-I data are reflected in the performance of the C-I&C-II-*dco* resources;

at 75% of all test sets around 20% of parse failures based on C-I&C-II-*dco* are due to unknown words. The results based on C-I&C-II-*ro* data are comparable.

7.4.3 Parse ambiguity

As mentioned in Section 7.3.2.4, the performance of the developed grammars was optimised for modelled utterances on per-dialogue basis. The number of parses in the development set ranged from 1 to 3, with most utterances yielding a single parse (see Figure 7.2). Now, higher generalisation power of the CCGs may come at a price of parse ambiguity. While in this work we do not address the problem of parse ranking or parse selection – identifying the most likely parse – we analyse the distributions of the number of parses on seen and unseen data in order to assess the complexity of the parse selection problem.

Parse ambiguity box plots are shown in Figure 7.6. On the seen data, the mean number of CCG parses is around one with a few outliers. The mean number of CFG parses is higher than the corresponding CCG results for C-II and C-I&C-II grammars when tested on C-I data. The performance of all grammars on C-II and C-I&C-II test sets is the same, one parse on average with a few outliers. The highest number of CCG parses is 6 and is found for a C-II utterance when parsed with C-II resources. The results show that even though ambiguity was tuned on per-dialogue basis, there is no dramatic increase in ambiguity when the complete lexicon is used.

The increase in parse ambiguity on unseen data is low; the number of CCG parses ranges from 1 to 7 (one outlier), by comparison with the 1 to 2 range of the CFG parser. The mean number of CCG parses remains between 1 and 2, negligibly higher than the CFG result. The 1 to 6 (seen data) or 7 (unseen data) range in the number of parses is manageable.

7.4.4 Overall performance of the deep parser

Finally, we look at the overall performance of the CCG parser based on *complete lexica*. Two measures are reported: the percentage of test set parsed and the oov error rates (summary of Tables 7.5, 7.6, 7.7, 7.8) and the percentage of proof utterances parsed per dialogue based on the combined C-I&C-II lexicon.

Table 7.9 summarises overall coverage of the final CCGs by test set. Combinations of development sets with obvious complete coverage results are marked with a dot.[15] On seen data, the C-II grammar parses almost the entire C-I development set (10% failures due to oov words) and thus also almost the entire combined set (3% oov). By contrast, the C-I grammar

[15] The *dco* and *ro* grammars are equivalent when the full lexicon is used.

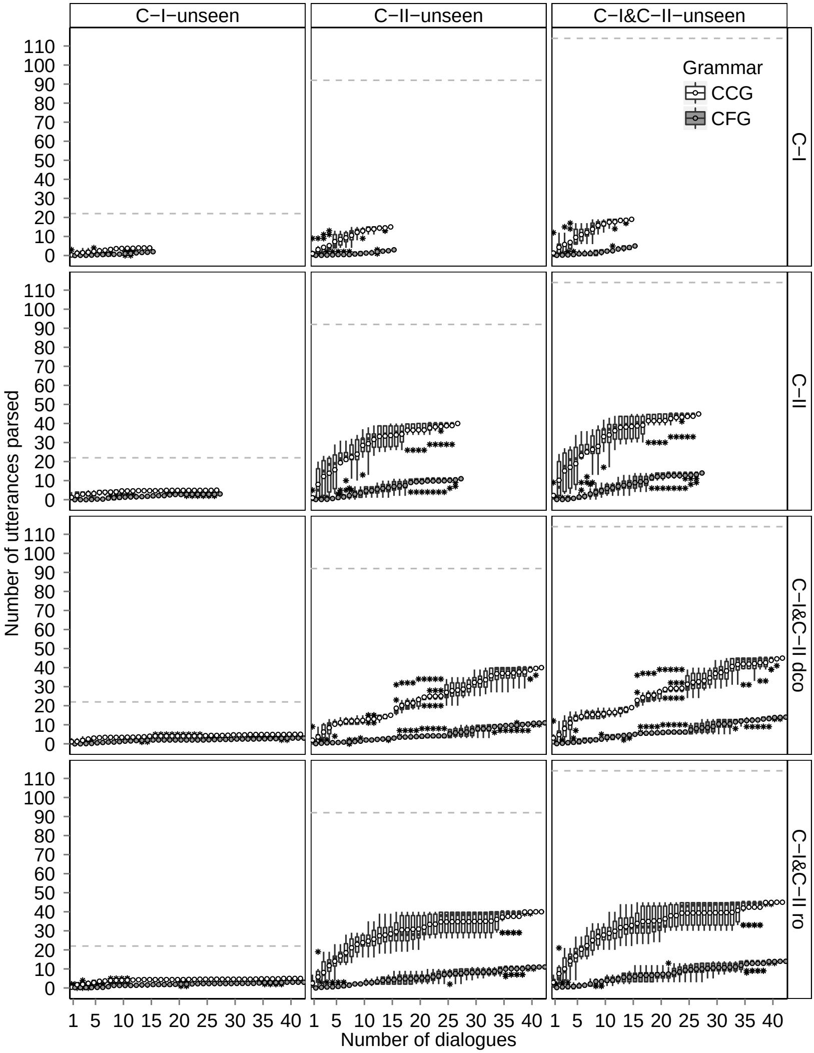


Figure 7.5: Growth of coverage on unseen data

Table 7.7: Mean coverage on the unseen data in percentage of test set parsed

Grammar development set		C-I-unseen (*N=22*)		C-II-unseen (*N=92*)		C-I&C-II-unseen (*N=114*)	
		CCG	CFG	CCG	CFG	CCG	CFG
C-I (n_d=*15*)	25%	**7.73**(6.12)	0.91(1.82)	**5.65**(3.85)	0.33(0.70)	**6.05**(4.10)	0.44(0.71)
	50%	**12.73**(3.96)	2.73(2.23)	**10.11**(3.44)	0.54(0.88)	**10.61**(3.15)	0.96(1.00)
	75%	**17.27**(1.82)	5.00(2.45)	**14.89**(1.46)	1.52(1.00)	**15.35**(1.48)	2.19(0.98)
	100%	**18.18**(0.00)	9.09(0.00)	**16.30**(0.00)	3.26(0.00)	**16.67**(0.00)	4.39(0.00)
C-II (n_d=*27*)	25%	**17.73**(4.29)	3.64(3.96)	**22.93**(8.34)	2.39(2.05)	**21.93**(7.24)	2.63(2.00)
	50%	**21.36**(2.08)	7.27(3.02)	**36.09**(5.95)	6.41(2.81)	**33.25**(5.17)	6.58(2.67)
	75%	**22.27**(1.36)	12.27(2.08)	**39.78**(4.46)	10.22(2.24)	**36.40**(3.83)	10.61(2.13)
	100%	**22.73**(0.00)	13.64(0.00)	**43.48**(0.00)	11.96(0.00)	**39.47**(0.00)	12.28(0.00)
C-I&C-II *dco* (n_d=*42*)	25%	**15.45**(2.23)	6.36(3.02)	**13.37**(1.62)	2.17(0.84)	**13.77**(1.67)	2.98(1.19)
	50%	**18.64**(1.36)	9.09(0.00)	**26.63**(3.71)	4.57(1.52)	**25.09**(3.24)	5.44(1.23)
	75%	**21.36**(2.08)	11.36(2.27)	**35.54**(5.01)	8.59(2.68)	**32.81**(4.39)	9.12(2.49)
	100%	**22.73**(0.00)	13.64(0.00)	**43.48**(0.00)	11.96(0.00)	**39.47**(0.00)	12.28(0.00)
C-I&C-II *ro* (n_d=*42*)	25%	**18.18**(2.03)	5.91(3.55)	**25.87**(4.53)	2.93(1.38)	**24.39**(3.82)	3.51(1.71)
	50%	**20.00**(2.23)	9.09(2.87)	**34.67**(6.21)	5.76(2.62)	**31.84**(5.44)	6.40(2.48)
	75%	**20.91**(2.23)	10.91(2.23)	**37.93**(6.20)	9.02(2.01)	**34.65**(5.43)	9.39(2.00)
	100%	**22.73**(0.00)	13.64(0.00)	**43.48**(0.00)	11.96(0.00)	**39.47**(0.00)	12.28(0.00)

Table 7.8: Mean proportion of parse failures due to oov words on unseen data

Grammar development set		C-I-unseen (*N=22*)	C-II-unseen (*N=92*)	C-I&C-II -unseen (*N=114*)
C-I (n_d=*15*)	25%	0.71 (0.10)	0.92 (0.04)	0.88 (0.05)
	50%	0.59 (0.08)	0.84 (0.05)	0.79 (0.05)
	75%	0.49 (0.04)	0.78 (0.02)	0.72 (0.02)
	100%	0.41 (0.00)	0.73 (0.00)	0.67 (0.00)
C-II (n_d=*27*)	25%	0.43 (0.13)	0.61 (0.16)	0.58 (0.15)
	50%	0.23 (0.11)	0.29 (0.13)	0.28 (0.12)
	75%	0.11 (0.09)	0.14 (0.12)	0.13 (0.11)
	100%	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
C-I&C-II *dco* (n_d=*42*)	25%	0.49 (0.05)	0.80 (0.03)	0.74 (0.03)
	50%	0.32 (0.07)	0.52 (0.08)	0.49 (0.08)
	75%	0.12 (0.11)	0.22 (0.11)	0.20 (0.11)
	100%	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
C-I&C-II *ro* (n_d=*42*)	25%	0.37 (0.12)	0.53 (0.09)	0.50 (0.10)
	50%	0.22 (0.10)	0.28 (0.15)	0.27 (0.13)
	75%	0.16 (0.07)	0.17 (0.09)	0.17 (0.09)
	100%	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)

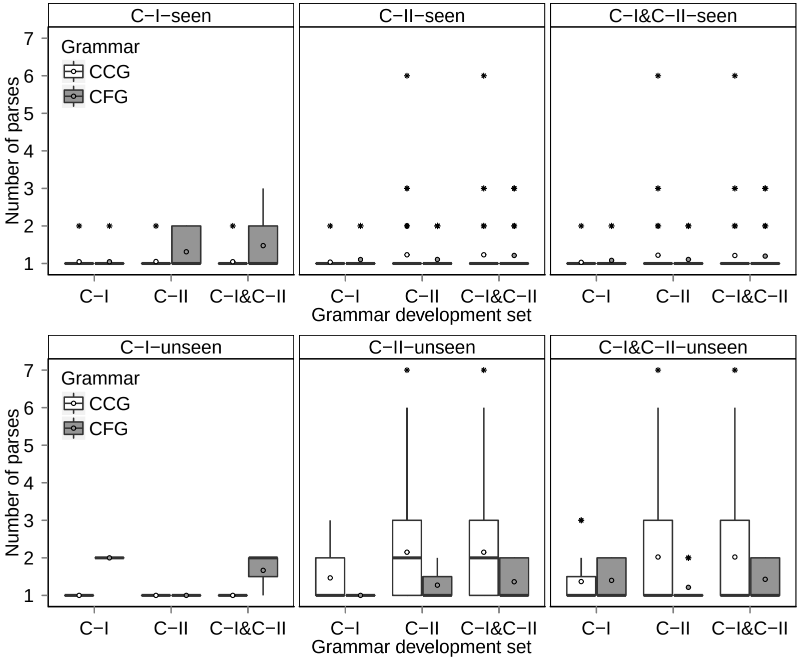


Figure 7.6: Parse ambiguity on seen (top) and unseen (bottom) data

accounts for merely 50% of C-II and the combined set. Around half of the parse failures are due to oov words. This shows that a lot of phenomena found in the binary relations corpus are not present in the set theory proofs, specifically also, C-II has a greater vocabulary size and the vocabulary is more diverse.

Performance drops dramatically on unseen data. The coverage of the C-I grammars remains below 20% for all test sets. 73% of the parse failures on unseen C-II data and 67% failures on unseen C-I&C-II data are due to oov errors; even on data stemming from the same corpus, the oov rate is high (41%). C-II grammars and the combined resources, C-I&C-II, account for barely over 20% of the unseen C-I and 40% of the unseen C-II utterances. Interestingly, C-I resources do not contribute to the coverage on the combined test set at all; results for C-II and C-I&C-II are the same. None of the unparsed utterances based on the combined grammars fail due to oov words since the unseen data set was built based on vocabulary found in the combined C-I&C-II development set. Interestingly, the 41% oov rate for C-I resources on C-I unseen set and the fact that C-II resources yield no failures due oov words suggest that the

Table 7.9: Summary of percentage coverage and oov rates on seen and unseen data with complete lexica (coverage/oov-failure-rate)

Grammar development set	C-I-seen (*N=21*)	C-II-seen (*N=56*)	C-I&C-II -seen (*N=61*)	C-I-unseen (*N=22*)	C-II-unseen (*N=92*)	C-I&C-II -unseen (*N=114*)
C-I (n_d =*15*)	.	48%/0.50	52%/0.46	18%/0.41	16%/0.73	17%/0.67
C-II (n_d =*27*)	90%/0.10	.	97%/0.03	23%/0.00	43%/0.00	39%/0.00
C-I&C-II *dco* / *ro* (n_d =*42*)	.	.	.	23%/0.00	43%/0.00	39%/0.00

C-I corpus is lexically more heterogeneous than the C-II corpus; some of the utterances in C-I-unseen must contain vocabulary not found in the modelled C-I utterances. This is not the case with the C-II data.

Figure 7.7 shows the histogram of percentage of proof utterances parsed per dialogue based on the full combined C-I&C-II grammar. *All* proof step utterances, both seen and unseen, are included. The data is binned in 20% intervals. Overall, per dialogue coverage is lower for C-I than for C-II. The majority of C-I dialogues are parsed at 40-60% coverage. By contrast, most of the C-II dialogues are parsed at at least 60% coverage (the majority at 60-80%). More of the C-I dialogues than of the C-II dialogues are completely parsed.

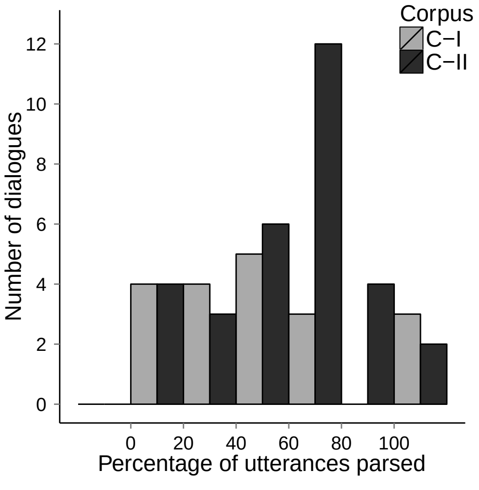


Figure 7.7: Histogram of percentages of proof utterances parsed per dialogue

7.5 Conclusions

The presented results let us draw conclusions along two dimensions: the potential of semantic grammars and the properties of the data. First, we have shown that hand-crafted semantic resources based on combinatory categorial grammars outperform baseline context-free grammars on the coverage measure while remaining at a manageable ambiguity level. Second, we have shown that the language used by students to talk about proofs is characterised by a large degree of diversity not only at a shallow level of wording patterns, but also at a deeper level of syntactic structures used.

The key conclusion we can draw is that the time overhead on the development of semantic grammars for students' proofs is *beneficial* and provided that more time is invested in data collection and grammar development, CCG as a grammar formalism has a potential of scaling well in this domain in spite of the unexpected diversity of the language. As previously mentioned, the coverage results point at the high linguistic diversity between the two corpora – thus between proofs in the two mathematical domains – manifested both at the lexical and syntactic level. (Recall that for the purpose of the experiments, vocabulary has been normalised with respect to domain-specific concept names. Thus, the lexical diversity within and between the corpora is not due to domain terminology.) Part of the reason for that might be that the binary relations problems were often solved using proof by cases whereas in set theory simple forward reasoning was most common. However, the most frequent statement type typical of proof by cases, assumption introduction, occurred in only 12 wording variants, of which only three appeared more than once ‘‘Sei...’’, ‘‘Sei nun...’’, and ‘‘Sei also...’’ (*Let..., Now, let..., Let then...*).

Finally, we believe that the data we have is insufficient, in the sense that it is not representative enough, for a *serious – robust –* proof tutoring system to be implemented *at the present stage*. The set of recurring verbalisations is small. This is against the intuition that the language of proofs should be small and repetitive. The set theory resources do not *yet* scale sufficiently even within-domain (C-I grammars tested on unseen data from the same corpus). The binary relations data scale better within-domain, however, across-domains (C-II resources tested on unseen C-I data) the difference in performance over within-domain data is negligible (23% vs. 18%: two utterances). More data would need to be collected. Interestingly, as a side-effect, our results give a little insight into the data collection methodology in the domain of proofs: Wizard-of-Oz experiments, logistically complex by themselves and in this case also cognitively demanding on the wizards, should cover multiple domains of mathematics rather than a single domain per experiment, as ours did, in order to provide more variety of proof verbalisations at one trial. Nevertheless, considering that the promising coverage *growth* results are based on 42 *partially* modelled dialogues, we also conclude that as far as language processing is concerned, natural language as the input mode for interactive proofs could be a matter of near future, provided that more data and human resources for grammar development were available. The question is though whether *typewritten* dialogue modality, in the times when spoken interaction with machines is becoming more and more widespread, mobile hand-held devices ubiquitous, and convenient graphical proof editors exist, whether *typewritten* dialogue with a proof tutoring system is what students would like to have.

Summary and outlook

This thesis contributes symbolic semantic processing methods for informal mathematical language, such as the language produced by students in interactions with a computer-based tutoring system for proofs. Unlike previous work on computational processing of textbook discourse, our work is grounded in systematic qualitative and quantitative corpus studies.

Students' language in computer-assisted proof tutoring The semantic processing approach we propose is motivated by a linguistic analysis of two corpora collected in experiments with a simulated system. Based on this data, we showed that students' language is rich in complex linguistic phenomena at the lexical, syntactic, semantic, and discourse-pragmatic level, and diverse in its verbalisation forms. Language production is moreover influenced by the presentation format of the study material. Material presented in natural language prompts verbosity in language production, whereas formalised presentation prompts dialogue contributions consisting mainly of formulas. This has practical implications for the implementation of tutorial dialogue systems for proofs and possibly also tutorial systems for mathematics in general. More natural language imposes more challenges on the input understanding component. In the context of mathematics, this necessitates reliable and robust parsing and discourse analysis strategies, including interpreting informal natural language interspersed with mathematical expressions. More symbolic language imposes stronger requirements on the mathematical expression parser since longer mathematical expressions tend to be prone to errors. Interestingly, our data suggests that students tend to have an informal attitude towards dialogue style while interacting with a tutoring system. This is manifested in the use of discourse markers typical of spoken language and suggests that students treat tutorial dialogue like a chat and adapt spoken language, which they would otherwise use when interacting with a human tutor, to the typewritten modality. Naturally, this makes the interaction even more informal and poses further challenges for input interpretation.

Semantic processing of informal mathematical language We have showed that mixed mathematical language consisting of natural language and symbolic notation lends itself well to syntactic analysis based on categorial grammars. Notation elements can be perspicuously modelled in terms of their syntactic categories and their semantic import can be thereby incorporated into the semantics of their natural language context. Previous computational approaches to textbook language either did not address the interactions between the two language ''modes'' at all or addressed it in a way which did not ensure generalisation.

A general language processing architecture for mathematical discourse which we proposed is parameterised for variables relevant to processing mathematical discourse in three scenarios (tutorial dialogue, mathematics assistance systems, and document processing) and modularised to facilitate portability. We proposed methods of modelling prominent syntactic and semantic language phenomena characteristic of informal mathematical proofs and of the German language of interaction specific to our data. The symbolic meaning representations generated by the parser have been shown to provide an appropriate level of semantic generalisation: we model semantic imprecision by providing a link between context-independent meaning and its context-specific interpretation through intermediate linguistically-motivated lexica and ontologies which enable interpretation of ambiguous wording and of a complex contextual operator. The intermediate knowledge representations have been shown to be relevant in modelling reference phenomena.

Prospects for natural language-based proof tutoring systems The performance of grammar resources developed based on corpus data has been evaluated in a simulation study. Manual development of linguistic resources for deep semantic processing is knowledge-intensive, time-consuming, and, consequently, costly; it requires familiarity with a linguistic formalism, both grammatical and semantic, and its computational implementation. Hand-crafted resources developed with a dedicated application in mind (often within a time-constrained project) tend to exhibit a serious lack of coverage beyond their specific domain. By contrast, wide-coverage hand-crafted resources, such as TRIPS (Allen, 1995) or even more so the ERG (Baldwin et al., 2004), are developed over many years and in collaboration with linguists. As shown by the LeActiveMath experiment, they do scale in a satisfactory way just by vocabulary adaptation (Callaway et al., 2006). We cannot expect a comparable coverage since our resources have been developed from scratch and based on minimally representative verbalisations in terms of frequency of occurrence. Nevertheless, the results we obtained show that categorial grammar as a basis of a parsing component, the critical step in a deep processing architecture, is a language model which provides better scalability in our domain than a simpler grammar formalism. This is an encouraging result and it implies that the language processing approach we propose is a viable contribution towards computational processing of informal mathematical language.

Outlook A fundamental question concerning the tutoring scenario within which this thesis has been set is the following: Is *typewritten* tutorial dialogue *the* proof tutoring method of the future? Although typewritten modality has been the state-of-the-art for most systems to date, it is somewhat hard to imagine a student typing to a proof tutoring system on his smart-phone or tablet; unless we consider a twitter-like dialogue, an idea possibly worth entertaining. This thesis offers processing methods suitable for contemporary systems and likely transferable to more advanced interfaces in which both typing and other modalities would be available. However, the way I see it plausible that interactive proof tutoring could evolve is towards *multi-modal input*. In multi-modal systems, formal proofs could be constructed via structured editors. Consider interfaces such as those of EPGY (McMath et al., 2001), ProofWeb (Hendriks et al., 2010), or the OpenProof (Barker-Plummer et al., 2008). Rigour and use of formal notation are the aspects of modern mathematics that sooner or later students need to learn. The formality of proof presentation in systems of this kind has another benefit: it makes the structure of proofs and the relations between statements explicit. Experience of a few semesters teaching mathematical logic let me think that this is what students actually prefer: say, Fitch-style deduction over proofs in prose. Natural language could be reserved for meta-level talk: students' questions, clarifications, requests for help and tutor's answers, explanations, and hints. *Spoken*, rather than written, *input modality* appears plausible, now that Nuance announced it's time for the *CUI*.[16] A WOz experiment would reveal the range of spoken verbalisations and help determine which language understanding methods would work. Now, the formal setup is unaccommodating with respect to students for whom formulas are an obstacle. Formalisation can be taught independently though and systems that teach translation to formal logic already exist; see, for instance, (Barwise and Etchemendy, 1999). These reflections lead me to concluding research on typewritten proof tutorial dialogue here.

This does not mean this thesis has no ‘‘further work’’. However, research building on this thesis shifts focus to mathematical prose. The trend towards open publishing has produced online repositories – so-called ‘‘digital

[16] *Beyond the GUI: It's Time for a Conversational User Interface* Ron Kaplan in *Wired*, 21. March 2013.

mathematical libraries’’ – many of which offer unlimited access to mathematical articles, and which open up possibilities for research on scholarly mathematical discourse. First, claims to the effect that mathematical language in narrative discourse should be repetitive, formulaic, and ‘‘small’’ should be verified by a systematic corpus analysis. My hypothesis is that these claims will not hold. Second, language processing methods proposed in this thesis will be evaluated on mathematical register language not only of proofs, but also other discourse types: definitions and theorems. Here, the ultimate goal is extraction of knowledge from mathematical documents. If proofs, definitions, and theorems are to be processed by deep grammars, as proposed here, a question arises of how to streamline the grammar development process. Our initial experiment based on a subset of dialogue data suggests that, in restricted domains, grammar engineering can be supported by an interactive process in which shallow similarity measures are used to cluster data, so that subsets of similar sentences are encoded in one step, thus making grammar engineering less prone to over-specialisation of lexical categories. We are presently setting up an experiment based on our entire dialogue corpus to evaluate the approach. Further, a known task in mathematics, akin to word-sense disambiguation, is the problem of determining the semantics of mathematical symbols in text. We have already made preliminary contributions in this domain (Grigore et al., 2009; Wolska and Grigore, 2010; Wolska et al., 2011) and we are planning to pursue this task further. In general though, what is obviously lacking in the state-of-the-art in processing mathematical discourse are basic language processing resources – annotated corpora – and components: sentence- and word-tokenisers, POS taggers, shallow parsers, named entity and domain term recognisers, the usual tools which in natural language processing are taken for granted. While this thesis ends my work on dialogue, there is a new niche to be filled that might come to be known as *MathNLP*.

Bibliography

Abel, A., B.-Y. Chang, and F. Pfenning. Human-Readable Machine-Verifiable Proofs For Teaching Constructive Logic. In *Proceedings of the IJCAR Workshop on Proof Transformations, Proof Presentations, and Complexity of Proofs*, 2001. 61

Abrahams, P. *Machine verification of mathematical proof.* PhD thesis, MIT, 1963. 44

Ajdukiewicz, K. Die syntaktische Konnexität. *Studia Philosophica*, 1935. 178

Aleven, V. and K. Koedinger. The Need for Tutorial Dialog to Support Self-Explanation. In *Proceedings of the AAAI Fall Symposium on Building Dialog Systems for Tutorial Applications*, 2000. 63

Alibert, D. and M. Thomas. Research on mathematical proof. In *Advanced Mathematical Thinking*. 1991. 29

Allen, J. *Natural Language Understanding*, 1995. 254, 278

Allen, J. and M. Core. Draft of DAMSL: Dialogue act markup in several layers, 1997. 39

Allen, J., B. Miller, E. Ringger, and T. Sikorski. A robust system for natural spoken dialogue. In *Proceedings of the ACL Conference*, 1996. 68

Almeida, D. A survey of mathematics undergraduates' interaction with proof: some implications for mathematics education. *International Journal of Mathematical Education in Science and Technology*, 2000. 29, 30

Alshawi, H. and R. Crouch. Monotonic semantic interpretation. In *Proceedings of the ACL Conference*, 1992. 178

Anderson, J. The legacy of school – attempts at justifying and proving among new undergraduates. *Teaching Mathematics and its Applications*, 1996. 29

Anderson, J., A. Corbett, K. Koedinger, and R. Pelletier. Cognitive tutors: Lessons learned. *Journal of the Learning Sciences*, 1995. 62

Anderson, R. Two-Dimensional Mathematical Notation. In *Syntactic Pattern Recognition Applications*. 1977. 97

Austin, J. *How to do things with words*, 1962. 39, 62

Autexier, S. and A. Fiedler. Textbook proofs meet formal logic – The problem of underspecification and granularity. In *Proceedings of the MKM Conference*, 2006. 41, 45, 143

Autexier, S., C. Benzmüller, A. Fiedler, H. Horacek, and B. Vo. Assertion-level Proof Representation with Under-Specification. In *Proceedings of the MKM Conference*, 2004. 41, 45

Autexier, S., C. Benzmüller, and J. Siekmann. OMEGA: Resource-Adaptive Processes in an Automated Reasoning System. In *Resource-Adaptive Cognitive Processes*. 2009. 42

Autexier, S., D. Dietrich, and M. Schiller. Towards an intelligent tutor for mathematical proofs. In *Proceedings of the Workshop on CTP Components for Educational Software*. 2012. 42, 167, 254

Bagchi, A. and C. Wells. Varieties of mathematical prose. *Problems, Resources and Issues in Mathematics Undergraduate Studies*, 1998. 31, 86

Bagni, G. Some Cognitive Difficulties Related to the Representations of two Major Concepts of Set Theory. *Educational Studies in Mathematics*, 2006. 110

Baker, E. and H. O'Neil, Eds. *Technology Assessment in Education and Training*, 1994. 253

Baldridge, J. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. PhD thesis, University of Edinburgh, 2002. 58, 194, 196, 209

Baldridge, J. and G.-J. Kruijff. Coupling CCG with Hybrid Logic Dependency Semantics. In *Proceedings of the ACL Conference*, 2002. 179, 197

Baldridge, J. and G.-J. Kruijff. Multi-Modal Combinatory Categorial Grammar. In *Proceedings of the EACL Conference*, 2003. 194, 197

Baldwin, T., E. Bender, D. Flickinger, A. Kim, and S. Oepen. Road-testing the English Resource Grammar over the British National Corpus. In *Proceedings of LREC*, 2004. 278

Bar-Hillel, Y. A Quasi-Arithmetical Notation for Syntactic Description. *Language*, 1953. 178

Barker-Plummer, D., J. Etchemendy, A. Liu, M. Murray, and N. Swoboda. Openproof: A flexible framework for heterogeneous reasoning. In *Proceedings of the Conference on Diagrammatic Representation and Inference*, 2008. 279

Baron, N. The Language of the Internet. In *Handbook for Language Engineers*. 2003. 157

Bartle, R. and D. Sherbert. *Introduction to Real Analysis*, 1982. 48, 96, 127, 146

Barwise, J. and J. Etchemendy. *Language, Proof and Logic*, 1999. 279

Baur, J. Syntax und Semantik mathematischer Texte. Diplomarbeit, Universität des Saarlandes, 1999. 48, 56

Becker, H. *Semantische und lexikalische Aspekte der mathematischen Fachsprache des 19. Jahrhunderts*. PhD thesis, Universität Oldenburg, 2006. 116

Becker, L., W. Ward, S. van Vuuren, and M. Palmer. DISCUSS: A dialogue move taxonomy layered over semantic representations. In *Proceedings of the IWCS*, 2011. 151, 154

Bell, A. A study of pupils' proof-explanations in mathematical situations. *Educational Studies in Mathematics*, 1976. 29

Benzmüller, C. and M. Kohlhase. LEO – A Higher-Order Theorem Prover. In *Proceedings of the CADE Conference*, 1998. 73

Benzmüller, C. and B. Vo. Mathematical domain reasoning tasks in natural language tutorial dialog on proofs. In *Proceedings of AAAI*, 2005. 40, 42, 45, 143, 144

Benzmüller, C., M. Jamnik, M. Kerber, and V. Sorge. Experiments with an agent-oriented reasoning system. In *Proceeding of the Joint German/Austrian KI Conference*, 2001. 73

Benzmüller, C., A. Fiedler, M. Gabsdil, H. Horacek, I. Kruijff-Korbayová, D. Tsovaltzi, B. Vo, and M. Wolska. A Wizard-of-Oz Experiment for Tutorial Dialogues in Mathematics. In *Proceedings of the Workshop on Advanced Technologies for Mathematics Education*, 2003a. 32, 61

Benzmüller, C., A. Fiedler, M. Gabsdil, H. Horacek, I. Kruijff-Korbayová, D. Tsovaltzi, B. Vo, and M. Wolska. Language phenomena in tutorial dialogs on mathematical proofs. In *Proceedings of the SemDial Workshop*, 2003b. 32, 61

Benzmüller, C., H. Horacek, I. Kruijff-Korbayová, H. Lesourd, M. Schiller, and M. Wolska. DiaWozII – A Tool for Wizard-of-Oz Experiments in Mathematics. In *Proceedings of the KI Conference*, 2006. 61, 65

Benzmüller, C., H. Horacek, H. Lesourd, I. Kruijff-Korbayová, M. Schiller, and M. Wolska. A corpus of tutorial dialogs on theorem proving; the influence of the presentation of the study-material. In *Proceedings of LREC*, 2006. 32, 61

Benzmüller, C., M. Schiller, and J. Siekmann. Resource-bounded Modelling and Analysis of Human-level Interactive Proofs. In *Resource-Adaptive Cognitive Processes*. 2009. 40, 42, 45

Bernsen, N., H. Dybkjær, and L. Dybkjær. *Designing Interactive Speech Systems – From First Ideas to User Testing*, 1998. 63

Biber, D. *Variation across speech and writing*, 1988. 157

Billingsley, W. and P. Robinson. An Interface for Student Proof Exercises Using MathsTiles Isabelle/HOL in an Intelligent Book. *Journal of Automated Reasoning*, 2007. 70

Blackburn, P. Representation, reasoning, and relational structures: a hybrid logic manifesto. *Logic Journal of the IGPL*, 2000. 179, 197

Bloom, B. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, 1984. 63

Blostein, D. and A. Grbavec. Recognition of mathematical notation. In *Handbook of Character Recognition and Document Image Analysis*. 1997. 68

Bobrow, D. *Natural language input for a computer problem solving system*. PhD thesis, MIT, 1964. 44

Bobrow, D., R. Kaplan, M. Kay, D. Norman, H. Thompson, and T. Winograd. GUS, A Frame-Driven Dialog System. *Artificial Intelligence*, 1977. 39

Booker, G. Valuing Language in Mathematics: Say what you mean and mean what you say. In *Valuing Mathematics in Society*, 2002. 30, 87, 100, 116

Borak, E. and A. Zalewska. Mizar Course in Logic and Set Theory. In *Towards Mechanized Mathematical Assistants*, 2007. 61

Bos, J., E. Mastenbroek, S. MacGlashan, S. Millies, and M. Pinkal. A compositional DRS-based formalism for NLP applications. Technical report, Universität des Saarlandes, 1994. 48

Botley, S., J. Glass, T. McEnery, and A. Wilson, Eds. *Approaches to Discourse Anaphora: Proceedings of the Discourse Anaphora and Anaphor Resolution Colloquium*, 1996. 234

Brennan, S. and J. Ohaeri. Effects of message style on users' attributions toward agents. In *Proceedings of Human Factors in Computing Systems Conference Companion*, 1994. 64

Bresnan, J. *Lexical Functional Syntax*, 2001. 178

Bronstein, I. and K. Semendjajew. *Taschenbuch der Mathematik*, Teubner, 1991. 77

Brown, C. Verifying and Invalidating Textbook Proofs using Scunak. In *Proceedings of the MKM Conference*, 2006a. 42

Brown, C. Scunak: User's Manual. Technical report, Universität des Saarlandes, 2006b. 42, 249

Brown, G. and G. Yule. *Discourse analysis*, 1983. 142

Buckley, M. *Modelling solution step discussions in tutorial dialogue*. PhD thesis, Universität des Saarlandes, 2010. 254

Buckley, M. and M. Wolska. Towards modelling and using common ground in tutorial dialogue. In *Proceedings of the SemDial Workshop*, 2007. 39, 144

Buckley, M. and M. Wolska. A Grounding Approach to Modelling Tutorial Dialogue Structures. In *Proceedings of the SemDial Workshop*, 2008a. 39

Buckley, M. and M. Wolska. A Classification of Dialogue Actions in Tutorial Dialogue. In *Proceedings of the COLING Conference*, 2008b. 154

Bunt, H. The DIT++ taxonomy for functional dialogue markup. In *Proceedings of the Workshop ''Towards a Standard Markup Language for Embodied Dialogue Acts''*. 2009. 39

Bunt, H., R. Morante, and S. Keizer. An empirically based computational model of grounding in dialogue. In *Proceedings of the SIGdial Meeting*, 2007. 39

Callaghan, P. and Z. Luo. Computer-Assisted Reasoning with Natural Language: Implementing a Mathematical Vernacular. In *Proceedings of the CLUK Research Colloquium*, 1997. 48

Callaway, C., M. Dzikovska, C. Matheson, J. Moore, and C. Zinn. Using dialogue to learn math in the LeActiveMath project. In *Proceedings of the Workshop on Language-enhanced Educational Technology*, 2006. 43, 70, 177, 254, 278

Campbell, G., N. Steinhauser, M. Dzikovska, J. Moore, C. Callaway, and E. Farrow. The DeMAND coding scheme: A ''common language'' for representing and analyzing student discourse. In *Proceedings of the AIED Conference*, 2009. 42, 151, 154

Carpenter, B. German Word Order and ''Linearization'' in Type-Logical Grammar. In *Proceedings of the Workshop on Current Topics in Constraint-based Theories of Germanic Syntax*, 1998. 196, 209

Carroll, J., T. Briscoe, and A. Sanfilippo. Parser Evaluation: a Survey and a New Proposal. In *Proceeding of LREC*, 1998. 254

Carroll, J., G. Minnen, and T. Briscoe. Parser evaluation: Using a grammatical relation annotation scheme. In *Treebanks: building and using parsed corpora*. 2003. 254

Chafe, W. Discourse structure and human knowledge. In *Language Comprehension and the Acquisition of Knowledge*. 1972. 136

Chafe, W. Givenness, Contrastiveness, Definiteness, Subjects, Topics, and Point of view. In *Subject and Topic*. 1976. 136

Chafe, W. and D. Tannen. The Relation between Written and Spoken Language. *Annual Review of Anthropology*, 1987. 157

Chaves, R. On the syntax and semantics of *vice versa*. In *Proceedings of the HPSG Conference*, 2010. 224

Chazan, D. High school geometry students' justifications for their views of empirical evidence and mathematical proof. *Educational Studies in Mathematics*, 1993. 29

Cinková, S., J. Hajič, M. Miluková, L. Mladová, A. Nedolužko, P. Pajas, J. Panevová, J. Semecký, J. Šindlerová, J. Toman, Z. Urešová, and Z. Žabokrtský. Annotation of English on the Tectogrammatical Level. Technical Report 35, Charles University, 2006. 192

Clark, H. Bridging. In *Theoretical Issues in Natural Language Processing*. 1975. 135

Clark, H. *Using Language*, 1996. 39, 63

Clark, H. and S. Brennan. Grounding in communication. In *Perspectives on Socially Shared Cognition*. 1991. 39

Clark, H. and E. Schaefer. Contributing to discourse. *Cognitive Science*, 1989. 39

Copestake, A., D. Flickinger, R. Malouf, S. Riehemann, and I. Sag. Translation using Minimal Recursion Semantics. In *Proceedings of the Conference on Theoretical and Methodological Issues in Machine Translation*, 1995. 178

Copestake, A., D. Flickinger, C. Pollard, and I. Sag. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 2005. 178

Copestake, A. *Implementing Typed Feature Structure Grammars*, 2001. 48

Croteau, E., N. Heffernan, , and K. Koedinger. Why are algebra word problems difficult? Using tutorial log files and the power law of learning to select the best fitting cognitive model. In *Proceedings of the ITS Conference*, 2004. 62

Crystal, D. *Language and the Internet*, 2001. 157

Dahl, D., Ed. *Practical Spoken Dialog Systems*, 2004. 62, 64

Dahlbäck, N., A. Jönsson, and L. Ahrenberg. Wizard of Oz studies: why and how. In *Proceedings of the Conference on Intelligent User Interfaces*, 1993. 63

Dahlbäck, N. and A. Jönsson. Empirical studies of discourse representation for natural language interfaces. In *Proceedings of the EACL Conference*, 1989. 67

Davis, P. and R. Hersh. *The Mathematical Experience*, 1981. 29

D'Mello, S., R. Picard, and A. Graesser. Toward an affect-sensitive AutoTutor. *IEEE Intelligent Systems*, 2007. 42

Dorier, J.-L., A. Robert, J. Robinet, and M. Rogalski. The obstacle of formalism in linear algebra. In *On the Teaching of Linear Algebra*. 2000. 30, 87, 104

Downs, M. and J. Mamona-Downs. The proof language as a regulator of rigor in proof, and its effect on student behaviour. In *Proceedings of the CERME 4*, 2005. 30, 87, 104

Dreyfus, T. Why Johnny Can't Prove. *Educational Studies in Mathematics*, 1999. 29

Dubinsky, E. and O. Yiparaki. On students' understanding of AE and EA quantification. In *Research in Collegiate Mathematics Education IV*. 2000. 134

Dzikovska, M., M. Swift, J. Allen, and W. de Beaumont. Generic parsing for multi-domain semantic interpretation. In *Proceedings of the Workshop on Parsing Technologies*, 2005. 254

Dzikovska, M., D. Reitter, J. Moore, and C. Zinn. Data-driven Modelling of Human Tutoring in Calculus. In *Proceedings of the Workshop on Language-enhanced Educational Technology*, 2006. 70

Dzikovska, M., C. Callaway, E. Farrow, M. Marques-Pita, C. Matheson, and J. Moore. Adaptive Tutorial Dialogue Systems Using Deep NLP Techniques. In *Proceedings of the NAACL Conference (Demo session)*, 2007. 42

Elsom-Cook, M. Student modelling in intelligent tutoring systems. *Artificial Intelligence Review*, 1993. 42

Epp, S. The Language of Quantification in Mathematics Instruction. In *Developing Mathematical Reasoning in Grages K-12*. 1999. 134

Ervynck, G. Mathematics as a foreign language. In *Proceedings of the Conference of the International Group for the Psychology of Mathematics Education*, 1992. 93

Evert, S. and M. Baroni. *zipfR*: Word Frequency Distributions in R. In *Proceedings of the ACL Conference*, 2007. 160

Fass, D. Metonymy and metaphor: what's the difference? In *Proceedings of the COLING Conference*, 1988. 243

Fateman, R. Handwriting + Speech for Computer Entry of Mathematics. `http://www.eecs.berkeley.edu/~fateman/papers/voice+hand.pdf` [Accessed: 2007], n.d.a. 70, 97, 101

Fateman, R. How can we speak math? `http.cs.berkeley.edu/~fateman/papers/speakmath.pdf` [Accessed: 2007], n.d.b. 68, 69, 97, 101

Ferreira, H. and D. Freitas. Enhancing the Accessibility of Mathematics for Blind People: The AudioMath Project. In *Proceedings of the Computers Helping People with Special Needs Conference*, 2004. 101

Ferreira, H. and D. Freitas. AudioMath – Towards Automatic Readings of Mathematical Expressions. In *Proceedings of the Human-Computer Interaction Conference*, 2005. 101

Fiedler, A. Natural Language Proof Presentation. In *Mechanizing Mathematical Reasoning: Essays in Honor of Jörg Siekmann on the Occasion of His 60th Birthday*. 2005. 43

Fiedler, A. and D. Tsovaltzi. Automating Hinting in Mathematical Tutorial Dialogue. In *Proceedings of the Workshop on Dialogue Systems: interaction, adaptation and styles of management*, 2003. 42

Fiedler, A., A. Franke, H. Horacek, M. Moschner, M. Pollet, and V. Sorge. Ontological Issues in the Representation and Presentation of Mathematical Concepts. In *Proceedings of the Workshop on Ontologies and Semantic Interoperability*, 2002. 146

Fiedler, A., M. Gabsdil, and H. Horacek. A Tool for Supporting Progressive Refinement of Wizard-of-Oz Experiments in Natural Language. In *Proceedings of the ITS Conference*, 2004. 65

Fine, K. A Defence of Arbitrary Objects. In *Proceedings of the Aristotelian Society*, 1983. 47

Fitzpatrick, D. Speaking technical documents: Using prosody to convey textual and mathematical material. In *Proceedings of the Computers Helping People with Special Needs Conference*, 2002. 101

Fitzpatrick, D. Mathematics: How and What to Speak. In *Proceedings of the Computers Helping People with Special Needs Conference*, 2006. 101

Forbes-Riley, K. and D. Litman. Adapting to student uncertainty improves tutoring dialogues. In *Proceedings of the AIED Conference*, 2009. 42

Fox, C. Vernacular Mathematics, Discourse Representation, and Arbitrary Objects. `http://csee.essex.ac.uk/staff/foxcj/papers/C-Fox-vernacular-paper.ps.gz` [Accessed: 2006], 1999. 47, 56

Francony, J., E. Kuijpers, and Y. Polity. Towards a methodology for Wizard of Oz experiments. In *Proceedings of the Conference on Applied Natural Language Proceeding*, 1992. 67

Frantzi, K., S. Ananiadou, and H. Mima. Automatic recognition of multi-word terms: the C-value/NC-value method. *International Journal on Digital Libraries*, 2000. 185

Fraser, B. A note on *vice versa*. *Linguistic Inquiry*, 1970. 128

Fraser, N. and G. Gilbert. Simulating speech systems. *Computer Speech & Language*, 1991. 63, 65

Fujimoto, M. and S. Watt. An Interface for Math e-Learning on Pen-Based Mobile Devices. In *Proceedings of the Mathematical User Interfaces Workshop*, 2010. 70

Fujimoto, M., T. Kanahori, and M. Suzuki. Infty Editor – A Mathematics Typesetting Tool With a Handwriting Interface and a Graphical Front-End to OpenXM Servers. *Computer Algebra: Algorithms, Implementations and Applications*, 2003. 70, 99

Galliers, J. and K. Spärck Jones. Evaluating natural language processing systems. Technical Report UCAM-CL-TR-291, University of Cambridge, 1993. 254

Ganesalingam, M. *The Language of Mathematics*. PhD thesis, Cambridge University, 2009. 47, 48, 56, 85, 138

Gergle, D., D. Millen, R. Kraut, and S. Fussell. Persistence matters: Making the most of chat in tightly coupled work. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2004. 68

Gerstenberger, C. and M. Wolska. Introducing Topological Field Information into CCG. In *Proceedings of the ESSLLI Student Session*, 2005. 32, 205

Gillan, D., P. Barraza, A. Karshmer, and S. Pazuchanics. Cognitive Analysis of Equation Reading: Application to the Development of the Math Genie. In *Proceedings of the Computers Helping People with Special Needs Conference*, 2004. 101

Gillman, L. *Writing Mathematics Well: A Manual for Authors*, 1987. 144

Ginsburg, H. The Clinical Interview in Psychological Research on Mathematical Thinking: Aims, Rationales, Techniques. *For the Learning of Mathematics*, 1981. 63

Goldson, D., S. Reeves, and R. Bornat. A Review of Several Programs for the Teaching of Logic. *The Computer Journal*, 1993. 61

Gould, J., J. Conti, and T. Hovanyecz. Composing letters with a simulated listening typewriter. *Communications of the ACM*, 1983. 64

Grefenstette, G. and P. Tapanainen. What is a word, what is a sentence? Problems of Tokenization. In *Proceedings of the Conference on Computational Lexicography and Text Research*, 1994. 146, 183, 184

Grice, H. Logic and conversation. In *Syntax and Semantics 3: Speech Acts*. 1975. 62, 142

Grigore, M., M. Wolska, and M. Kohlhase. Towards Context-Based Disambiguation of Mathematical Expressions. In *Proceedings of the Joint Conference of ASCM and MACIS*, 2009. 280

Grishman, R. and R. Kittredge, Eds. *Analyzing Language in Restricted Domains: Sublanguage Description and Processing*, 1986. 86

Grishman, R., C. Macleod, and J. Sterling. Evaluating Parsing Strategies using Standardized Parse Files. In *Proceedings of the Conference on Applied Natural Language Processing*, 1992. 254

Grosz, B. Discourse analysis. In *Understanding Spoken Language*. 1978. 63

Grottke, S., S. Jeschke, N. Natho, S. Rittau, and R. Seiler. mArachna: Automated creation of knowledge representations for mathematics. In *Proceedings of the Interactive Computer-aided Learning Conference*, 2005a. 50

Grottke, S., S. Jeschke, N. Natho, S. Rittau, and R. Seiler. mArachna: Entwicklung von wissensrepräsentationsmechanismen für die mathematik. In *Proceedings of the Leipziger Informatik-Tage*, 2005b. 51

Grottke, S., S. Jeschke, N. Natho, and R. Seiler. mArachna: A Classification Scheme for Semantic Retrieval in eLearning Environments in Mathematics. In *Proceedings of the Conference on Multimedia and Information & Communication Technologies in Education*, 2005c. 50

Gruber, T. and G. Olsen. An Ontology for Engineering Mathematics. In *Proceedings of the Principles of Knowledge Representation and Reasoning Conference*, 1994. 219

Guy, C., M. Jurka, S. Stanek, and R. Fateman. Math Speak & Write, a Computer Program to Read and Hear Mathematical Input, 2004. `http://www.cs.berkeley.edu/~fateman/msw/AcademicPaper.pdf` [Accessed: 2006]. 69, 101

Hajičová, E. Theoretical description of language as a basis of corpus annotation: The case of Prague Dependency Treebank. In *Prague Linguistic Circle Papers*. 2002. 190

Hajičová, E., J. Panevová, and P. Sgall. A Manual for Tectogrammatical Tagging of the Prague Dependency Treebank. Technical report, Charles University, 2000. 190, 192

Hall, R. An analysis of errors made in the solution of simple linear equations. *Philosophy of Mathematics Education Journal*, 2002. 107

Hallgren, T. and A. Ranta. An Extensible Proof Text Editor. In *Logic For Programming and Automated Reasoning*, 2000. 48

Halmos, P. How to write mathematics. *L'Énseignement Mathématique*, 1970. 31, 86, 126, 143, 144

Hanna, G. Some pedagogical aspects of proof. *Interchange*, 1990. 142

Hanna, G. Challenges to the importance of proof. *For the Learning of Mathematics*, 1995. 29

Hanna, G. Proof, explanation and exploration: An overview. *Educational Studies in Mathematics*, 2000. 29

Hardy, G. and E. Wright. *An introduction to the theory of numbers*, 4th edition, 1971. 48

Harris, Z. *Mathematical structures of language*, 1968. 86

Hawkins, J. *Definiteness and Indefiniteness: A Study in Reference and Grammaticality Prediction*, 1978. 136

Heffernan, N. and K. Koedinger. Intelligent tutoring systems are missing the tutor: Building a more strategic dialog-based tutor. In *Proceedings of the AAAI Fall Symposium on Building Dialog Systems for Tutorial Applications*, 2000. 63

Heffernan, N. and K. Koedinger. An Intelligent Tutoring System Incorporating a Model of an Experienced Human Tutor. In *Proceedings of the ITS Conference*, 2002. 68

Heid, M. and M. Edwards. Computer algebra systems: Revolutions or retrofit for today's mathematics classroom? *Theory into Practice: Realizing Reform in School Mathematics*, 2001. 61

Hendriks, M., C. Kaliszyk, F. van Raamsdonk, and F. Wiedijk. Teaching logic using a state-of-the-art proof assistant. *Acta Didactica Napocensia*, 2010. 61, 279

Hepple, M. *The Grammar and Processing of Order and Dependency: a Categorial Approach*. PhD thesis, University of Edinburgh, 1990. 196

Herring, S. Interactional Coherence in CMC. *Journal of Computer-Mediated Communication*, 1999. 68

Hersh, R. Proving is convincing and explaining. *Educational Studies in Mathematics*, 1993. 142

Hersh, R. Prove – Once more and again. *Philosophia Mathematica*, 1997a. 29

Hersh, R. *What Is Mathematics,* Really*?*, 1997b. 30

Hildebrandt, B., H.-J. Eikmeyer, G. Rickheit, and P. Weiß. Inkrementelle Sprachrezeption. In *Proceedings der Fachtagung der Fachtagung der Gesellschaft für Kognitionswissenschaft*. 1999. 209

Hirschman, L. and N. Sager. Automatic Information Formatting of a Medical Sublanguage. In *Sublanguage: Studies of Language in Restricted Semantic Domains*. 1982. 86

Hobbs, J. Granularity. In *Proceedings of IJCAI*, 1985. 143

Hockenmaier, J. Creating a CCGbank and Wide-Coverage CCG Lexicon for German. In *Proceedings of the Joint COLING/ACL Conference*, 2006. 196

Höhle, T. Topologische felder. Technical report, Universität Köln, 1983. 207

Holland-Minkley, A., R. Barzilay, and R. Constable. Verbalization of high-level formal proofs. In *Proceedings of AAAI/IAAI*, 1999. 43

Horacek, H. and M. Wolska. Interpretation of Implicit Parallel Structures. A Case Study with ''vice-versa''. In *Proceedings of the NLDB Conference*, 2005a. 32, 205

Horacek, H. and M. Wolska. Fault-Tolerant Context-Based Interpretation of Mathematical Formulas. In *Proceedings of IJCAI*, 2005b. 107

Horacek, H. and M. Wolska. Interpretation of mixed language input in a mathematics tutoring system. In *Proceedings of the Workshop on Mixed Language Explanations in Learning Environments*, 2005c. 32, 205

Horacek, H. and M. Wolska. A Hybrid Model for Tutorial Dialogs. In *Proceedings of the SIGdial Meeting*, 2005d. 39

Horacek, H. and M. Wolska. Interpreting semi-formal utterances in dialogs about mathematical proofs. *Data and Knowledge Engineering Journal*, 2006a. 32, 205

Horacek, H. and M. Wolska. Handling errors in mathematical formulas. In *Proceedings of the ITS Conference*, 2006b. 32, 107, 205

Horacek, H. and M. Wolska. Transformation-based Interpretation of Implicit Parallel Structures: Reconstructing the meaning of ''vice versa'' and similar linguistic operators. In *Proceedings of the Joint COLING/ACL Conference*, 2006c. 32, 205

Horacek, H. and M. Wolska. Generating responses to formally flawed problem-solving statements. In *Proceedings of the AIED Conference*, 2007. 107, 173, 232

Horacek, H. and M. Wolska. Addressing Formally-Flawed Mathematical Formulas in Tutorial Dialogs. Error Analysis for Supporting Informed Reactions. In *Proceedings of the European Meeting on Cybernetics and Systems Research*, 2008. 107, 173

Horacek, H. Expressing references to rules in proof presentations. In *Proceedings of the Conference on Automated Reasoning*, 2001a. 43

Horacek, H. Ontological Aspects in Representing Mathematical Knowledge for Reasoning and Presentation Purposes. In *Proceedings of the KI Workshop on Ontologies*, 2001b. 219

Horacek, H., A. Fiedler, A. Franke, M. Moschner, M. Pollet, and V. Sorge. Representation of Mathematical Concepts for Inferencing and for Presentation Purposes. In *Proceedings of the European Meeting on Cybernetics and Systems Research*, 2004. 146, 199, 219

Hård af Segerstad, Y. *Use and Adaptation of Written Language to the Conditions of Computer-Mediated Communication*. PhD thesis, University of Gothenburg, 2002. 157, 174

Huang, X. and A. Fiedler. Proof Verbalization as an Application of NLG. In *Proceedings of IJCAI*, 1997. 43

Humayoun, M. and C. Raffalli. MathNat – Mathematical Text in a Controlled Natural Language. In *Proceedings of the CICLing Conference*, 2010. 52

Isaacs, E. and H. Clark. References in conversation between experts and novices. *Journal of Experimental Psychology*, 1987. 39

Jansen, A., K. Marriott, and G. Yelland. Perceiving structure in mathematical expressions. In *Proceedings of the Conference of the Cognitive Science Society*, 1999. 98, 138

Jansen, A., K. Marriott, and G. Yelland. Constituent Structure in Mathematical Expressions. In *Proceedings of the Conference of the Cognitive Science Society*, 2000. 98, 138

Jansen, A., K. Marriott, and G. Yelland. Comprehension of algebraic expressions by experienced users of mathematics. *The Quaterly Journal of Experimental Psychology*, 2003. 98, 138

Jeschke, S., N. Natho, S. Rittau, and M. Wilke. mArachna – automatically extracting ontologies from mathematical natural language texts. In *Proceedings of the Multiconference of Engineers and Computer Scientists*, 2007a. 51, 58

Jeschke, S., N. Natho, and M. Wilke. mArachna – automatic generation of mathematical ontologies from natural language texts. In *Proceedings of the Conference on Computer Aided Blended Learning*, 2007b. 51, 58

Jeschke, S., M. Wilke, N. Natho, and O. Pfeiffer. Managing mathematical texts with OWL and their graphical representation. In *Proceedings of the Hawaii Conference on System Sciences*, 2008. 50, 51, 176, 177, 186

Kamareddine, F. and R. Nederpelt. A refinement of de bruijn's formal language of mathematics. *Journal of Logic Language and Information*, 2004. 53

Kamareddine, F. and J. Wells. MathLang: A new language for mathematics, logic, and proof checking. (Research proposal), 2001. 53

Kamareddine, F. and J. Wells. Computerizing Mathematical Text with MathLang. In *Proceedings of the Workshop on Logical and Semantic Frameworks, with Applications*, 2008. 53

Kamareddine, F., M. Maarek, and J. Wells. Toward Object-Oriented Structure for Mathematical Text. In *Proceedings of the MKM Conference*, 2006. 54

Kamareddine, F., R. Lamar, M. Maarek, and J. Wells. Restoring Natural Language as a Computerised Mathematics Input Method. In *Proceedings of the MKM Conference*. 2007a. 54

Kamareddine, F., M. Maarek, K. Retel, and J. Wells. Gradual Computerisation/Formalisation of Mathematical Texts into Mizar. *Studies in Logic, Grammar and Rhetoric*, 2007b. 54

Kamareddine, F., J. Wells, and C. Zengler. Computerising Mathematical Text with MathLang, n.d. http://www.cedar-forest.org/forest/papers/drafts/mathlang-coq-short.pdf [Accessed: 2009]. 54

Kane, R., M. Byrne, and M. Hater. *Helping Children Read Mathematics*, 1974. 116

Kapitan, T. The ontological significance of variables. *Metaphysica*, 2002. 134

Karagjosova, E. Marked Informationally Redundant Utterances in Tutorial Dialogue. In *Proceedings of the SemDial Workshop*, 2003. 144

Karshmer, A. and D. Gillan. How well can we read equations to blind mathematics students: some answers from psychology. In *Proceedings of the HCI Conference*, 2003. 101

Kay, P. Contextual Operators: respective, respectively, and vice versa. In *Proceedings of the Berkeley Linguistics Society Meeting*, 1989. 128

Kelley, J. An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Information Systems*, 1984. 64, 65

Kelly, A. and R. Lesh, Eds. *Handbook of research design in mathematics and science education*, 2000. 61, 63

Kennedy, G., J. Eliot, and G. Krulee. Error patterns in problem solving formulations. *Psychology in the Schools*, 1970. 107

Kieran, C. Concepts associated with the equality symbol. *Educational Studies in Mathematics*, 1981. 110

Kirshner, D. *The Grammar of Symbolic Elementary Algebra*. PhD thesis, University of British Columbia, 1987. 98

Kiss, T. and J. Strunk. Unsupervised Multilingual Sentence Boundary Detection. *Computational Linguistics*, 2006. 146, 183

Kittredge, R. and J. Lehrberger, Eds. *Sublanguage: Studies of Language in Restricted Semantic Domains*, 1982. 86

Knuth, D., T. Larrabee, and R. Roberts. *Mathematical Writing*, 1989. 48, 86

Knuth, D. *The TEX Book*, (Also: http://www.latex-project.org [Accessed: 2006]), 1986. 182

Knuth, E. Secondary School Mathematics Teachers' Conceptions of Proof. *Journal for Research in Mathematics Education*, 2002. 29

Knuth, E., M. Alibali, N. McNeil, A. Weinberg, and A. Stephens. Middle school students' understanding of core algebraic concepts: Equivalence & Variable. *Zentralblatt für Didaktik der Mathematik*, 2005. 110

Koedinger, K., J. Anderson, W. Hadley, and M. Mark. Intelligent Tutoring Goes To School in the Big City. *Journal of Artificial Intelligence in Education*, 1997. 62

Koehn, P. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of MT Summit*, 2005. 224, 231

Kohlhase, A. and M. Kohlhase. Communities of Practice in MKM: An Extensional Model. In *Proceedings of the MKM Conference*, 2006. 105

Kohlhase, M. *OMDoc – An Open Markup Format for Mathematical Documents* (v1.2), 2006. 53

Kohlhase, M. and A. Franke. MBase: Representing Knowledge and Context for the Integration of Mathematical Software Systems. *Journal of Symbolic Computation*, 2001. 146, 199

Kubo, J., K. Tsuji, and S. Sugimoto. Automatic Term Recognition based on the Statistical Differences of Relative Frequencies in Different Corpora. In *Proceedings of LREC*, 2010. 185

Kvasz, L. The history of algebra and the development of the form of its language. *Philosophia Mathematica*, 2006. 92

Lakoff, G. and R. Núñez. *Where mathematics comes from: How the Embodied Mind Brings Mathematics into Being*, 2000. 126

Lamar, R. *A Partial Translation Path from MathLang to Isabelle*. PhD thesis, Heriot-Watt University, 2011. 54

Lambek, J. The mathematics of sentence structure. *The American Mathematical Monthly*, 1958. 178, 194

Leiser, R. Exploiting convergence to improve natural language understanding. *Interacting with Computers*, 1989. 64

Lemon, O. and X. Liu. DUDE: a dialogue and understanding development environment, mapping business process models to information state update dialogue systems. In *Proceedings of the EACL Conference*, 2006. Posters & Demonstrations. 39

Li, S., B. Wrede, and G. Sagerer. A computational model of multi-modal grounding for human-robot interaction. In *Proceedings of the SIGdial Meeting*, Sydney, Australia, 2006. 39

Libbrecht, P. Notations around the world: census and exploitation. In *Proceedings of the MKM Conference*, 2010. 105

Linebarger, M., D. Dahl, L. Hirschman, and R. Passoneau. Sentence fragments regular structure. In *Proceedings of the ACL Conference*, 1988. 86

Litman, D. and K. Forbes-Riley. Predicting student emotions in computer-human tutoring dialogues. In *Proceedings of the ACL Conference*, 2004. 42

Litman, D. and S. Silliman. ITSPOKE: An intelligent tutoring spoken dialogue system. In *Proceedings of the HLT/NAACL Conference*, 2004. 68

Litman, D., C. Rosé, K. Forbes-Riley, K. VanLehn, D. Bhembe, and S. Silliman. Spoken versus typed human and computer dialogue tutoring. *Journal of Artificial Intelligence in Education*, 2005. 69

Loper, E. and S. Bird. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics*, 2002. 261

Magne, O. Literature on Special Educational Needs in Mathematics: A bibliography with some comments. Technical report, Malmö University, 2001. 107

Marineau, J., P. Wiemer-Hastings, D. Harter, B. Olde, P. Chipman, A. Karnavat, V. Pomeroy, S. Rajan, and A. Graesser. Classification of Speech Acts in Tutorial Dialogue. In *Proceedings of the Workshop on Modeling Human Teaching Tactics and Strategies*, 2000. 42, 151, 154

Mark, M. and J. Greer. Evaluation Methodologies for Intelligent Tutoring Systems. *Journal of Artificial Intelligence in Education*, 1993. 253

Matheson, C., M. Poesio, and D. Traum. Modelling grounding and discourse obligations using update rules. In *Proceedings of the NAACL Conference*, 2000. 39

Matsuda, N. and K. VanLehn. Advanced Geometry Tutor: An intelligent tutor that teaches proof-writing with construction. In *Proceedings of the AIED Conference*, 2005. 62

Maynor, N. The Language of Electronic Mail: Written Speech? In *Centennial Usage Studies*. 1994. 157

McCawley, J. On the Applicability of Vice Versa. *Linguistic Inquiry*, 1970. 128

McConville, M. *An Inheritance-Based Theory of the Lexicon in Combinatory Categorial Grammar*. PhD thesis, University of Edinburgh, 2007. 196

McDonald, J. The EXCHECK CAI system. In *University-level computer-assisted instruction at Stanford: 1968–1980*. 1981. 46

McMath, D., M. Rozenfeld, and R. Sommer. A Computer Environment for Writing Ordinary Mathematical Proofs. In *Proceedings of the Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, 2001. 45, 70, 279

McTear, M. Modelling spoken dialogues with state transition diagrams: experiences with the CSLU toolkit. In *Proceedings of the Conference on Spoken Language Processing*, 1998. 39

McTear, M. *Spoken Dialogue Technology – Toward the Conversational User Interface*, 2004. 62, 64

Melis, E. Erroneous examples as a source of learning in mathematics. In *Proceedings of the Conference on Cognition and Exploratory Learning in the Digital Age*, 2004. 107

Melis, E., E. Andres, J. Budenbender, A. Frischauf, G. Goduadze, P. Libbrecht, M. Pollet, and C. Ullrich. ActiveMath: A generic and adaptive web-based learning environment. *Journal of Artificial Intelligence in Education*, 2001. 43, 61

Melis, E., J. Haywood, and T. Smith. LeActiveMath. In *Innovative Approaches for Learning and Knowledge Sharing*. 2006. 43, 61

Merrill, D., B. Reiser, M. Ranney, and J. Trafton. Effective Tutoring Techniques: A Comparison of Human Tutors and Intelligent Tutoring Systems. *Journal of Learning Sciences*, 1992. 63

Metzing, D. ATNS used as a procedural dialog model. In *Proceedings of the COLING Conference*, 1980. 39

Michael, J., A. Rovick, M. Glass, Y. Zhou, and M. Evens. Learning from a computer tutor with natural language capabilities. *Interactive Learning Environments*, 2003. 68

Michener, E. R. Understanding understanding mathematics. *Cognitive Science*, 1978. 29

Mikheev, A. Tagging sentence boundaries. In *Proceedings of the NAACL Conference*, 2000. 183

Mikheev, A. Period, Capitalized Words, etc. *Computational Linguistics*, 2002. 146

Mitkov, R. Pronoun resolution: The practical alternative. *Corpus-based and Computational Approaches to Discourse Anaphora*, 2000. 234

Mollá, D. and B. Hutchinson. Intrinsic versus extrinsic evaluations of parsing systems. In *Proceedings of the Workshop on Evaluation Initiatives in NLP: are evaluation methods, metrics and resources reusable?*, 2003. 254

Moore, J. What makes human explanations effective? In *Proceedings of the Conference of the Cognitive Science Society*, 1993. 30

Moore, R. C. Making the transition to the formal proof. *Educational Studies in Mathematics*, 1994. 29, 30, 87, 104

Moortgat, M. *Categorical Investigations: Logical and Linguistic Aspects of the Lambek Calculus*, 1988. 194

Morel, M. Computer–human communication. In *The Structure of Multimodal Dialogue*. 1989. 63, 69

Mostow, J. and G. Aist. Evaluating tutors that listen: An overview of project listen. In *Smart Machines in Education: The coming revolution in educational technology*. 2001. 68

Müller, S. *Deutsche Syntax deklarativ. Head-Driven Phrase Structure Grammar für das Deutsche*, 1999. 51, 209

Müller, S. Mehrfache Vorfeldbesetzung. *Deutsche Sprache*, 2003. 207, 209

Natho, N. *mArachna: eine semantische Analyse der mathematischen Sprache für ein computergestütztes Information Retrieval System*. PhD thesis, Technische Universität Berlin, 2005. 48, 50, 51, 85, 124, 145, 149

Natho, N., S. Jeschke, O. Pfeiffer, and M. Wilke. Natural Language Processing Methods for Extracting Information from Mathematical Texts. In *Advances in Communication Systems and Electrical Engineering*. 2008. 50, 51, 145

Nederpelt, R. and F. Kamareddine. Formalising the natural language of mathematics: A mathematical vernacular. In *Proceedings of the Tbilisi Symposium on Language, Logic, and Computation*, 2001. 53

O'Malley, M., D. Kloker, and B. Dara-Abrams. Recovering parentheses from spoken algebraic expressions. *IEEE Transactions on Audio and Electroacoustics*, 1973. 101

Palmer, D. and M. Hearst. Adaptive multilingual sentence boundary disambiguation. *Computational Linguistics*, 1997. 183

Pappuswamy, U., P. Jordan, and K. VanLehn. Resolving discourse deictic anaphors in tutorial dialogues. In *Proceedings of the Constraints in Discourse Workshop*, 2005. 234, 235

Pazienza, M., M. Pennacchiotti, and F. M. Zanzotto. Terminology extraction: An analysis of linguistic and statistical approaches. In *Knowledge Mining*. 2005. 185

Person, N. and A. Graesser. Fourteen facts about human tutoring: Food for thought for ITS developers. In *Proccedings of the Workshop on Tutorial Dialogue Systems: With a View Towards the Classroom*, 2003. 63

Piaget, J. *The Equilibrium of Cognitive Structures*, 1985. 88

Pinkal, M., J. Siekmann, C. Benzmüller, and I. Kruijff-Korbayová. DIALOG: Natual Language-based Interaction with a Mathematics Assistance System. In *SFB 378: Resource-adaptive Cognitive Processes*. 2004. 33

Pinkal, M., J. Siekmann, and C. Benzmüller. DIALOG: Tutorieller Dialog mit einem mathematischen Assistenzsystem. In *SFB 378: Ressourcenadaptive kognitive Prozesse*. 2001. 33

Pirker, H., G. Loderer, and H. Trost. Thus Spoke the User to the Wizard. In *Proceedings of the Eurospeech Conference*, 1999. 63

Poesio, M., A. Patel, and B. Di Eugenio. Discourse Structure and Anaphora in Tutorial Dialogues: An Empirical Analysis of Two Theories of the Global Focus. *Research on Language and Computation*, 2006. 234

Poesio, M., S. Ponzetto, and Y. Versley. Computational Models of Anaphora Resolution: A Survey, 2010. Online draft. 234

Pollard, C. and I. Sag. *Head-Driven Phrase Structure Grammar*, 1994. 48, 178

Pon-Barry, H., B. Clark, E. Bratt, K. Schultz, and S. Peters. Evaluating the effectiveness of SCoT: A spoken conversational tutor. In *Proceedings of the Workshop on Dialogue-based Intelligent Tutoring Systems*, 2004. 69

Pontelli, E., A. Karshmer, and G. Gupta. Mathematics and Accessibility: A Survey. In *The Universal Access Handbook*. 2009. 101

Popescu, O. and K. Koedinger. Towards understanding geometry explanations. In *Proceedings of the AAAI Fall Symposium on Building Dialogue Systems for Tutorial Applications*, 2000. 62

Porayska-Pomsta, K., C. Mellish, and H. Pain. Aspects of speech act categorisation: Towards generating teachers' language. *Journal of Artificial Intelligence in Education*, 2000. 42

Porayska-Pomsta, K., M. Mavrikis, and H. Pain. Diagnosing and acting on student affect: the tutor's perspective. *User Modeling and User-Adapted Interaction*, 2008. 42

Prince, E. Toward a Taxonomy of Given-New Information. In *Radical Pragmatics*. 1981. 136

Raiker, A. Spoken Language and Mathematics. *Cambridge Journal of Education*, 2002. 116

Raman, T. V. *Audio System for Technical Readings*. PhD thesis, Cornell University, 1994. 68, 101

Raman, T. V. *Auditory User Interfaces: Toward the Speaking Computer*, 1997. 68, 101

Raman, T. V. ASTER – Towards Modality-Independent Electronic Documents. *Multimedia Tools and Applications*, 1998. 68

Ranta, A. Type theory and the informal language of mathematics. In *Types for Proofs and Programs*. 1994. 48

Ranta, A. *Type-Theoretical Grammar*, 1995a. 48

Ranta, A. Syntactic categories in the language of mathematics. In *Types for Proofs and Programs*. 1995b. 48

Ranta, A. Context-relative syntactic categories and the formalization of mathematical text. In *Types for Proofs and Programs*. 1996. 48

Ravaglia, R., T. Alper, M. Rozenfeld, and P. Suppes. Successful pedagogical applications of symbolic computation. In *Computer-Human Interaction in Symbolic Computation*. 1999a. 61, 73

Ravaglia, R., R. Sommer, M. Sanders, G. Oas, and C. DeLeone. Computer-based Mathematics and Physics for Gifted Remote Students. In *Proceedings of the Conference on Mathematics/Science Education and Technology*, 1999b. 61

Reichman, R. *Getting Computers to Talk Like You and Me*, 1985. 63

Reiter, E. and R. Dale. *Building Natural Language Generation Systems*, 2000. 42

Reynar, J. and A. Ratnaparkhi. A maximun entropy approach to identifying sentence boundaries. In *Proceedings of the ANLP Conference*, 1997. 146, 183

Richards, M. and K. Underwood. Talking to machines: How are people naturally inclined to speak? *Contemporary Ergonomics*, 1984. 63, 69

Ringle, M. and R. Halstead-Nussloch. Shaping user input: a strategy for natural language dialogue design. *Interacting with Computers*, 1989. 64

Rips, L. *The psychology of proof: Deduction in human thinking*, 1994. 143

Rosé, C. and R. Freedman, Eds. *Building Dialog Systems for Tutorial Applications – Papers from the AAAI Fall Symposium*, 2000. 68

Rosé, C. and C. Torrey. Interactivity and expectation: Eliciting learning oriented behaviour with tutorial dialogue systems. In *Proceedings of the HCI Conference*, 2005. 63

Rosé, C., P. Jordan, P. Ringenberg, M. Siler, K. VanLehn, and A. Weinstein. Interactive conceptual tutoring in Atlas-Andes. In *Proceedings of the AIED Conference*, 2001. 42

Rudnicki, P. An overview of the MIZAR project. In *Selected Papers from the Types for Proofs and Programs Workshop*, 1992. 52

Rudnicky, A. Mode preference in a simple data-retrieval task. In *Proceedings of the HLT Workshop*, 1993. 68

Sáenz-Ludlow, A. and C. Walgamuth. Third Grader's Interpretations of Equality and the Equal Symbol. *Educational Studies in Mathematics*, 1998. 110

Sager, N. Syntactic formatting of science information. In *AFIPS Conference Proceedings*, 1972. Reprinted in *Sublanguage: Studies of Language in Restricted Semantic Domains*. 86

Salber, D. and J. Coutaz. Applying the Wizard of Oz Technique to the Study of Multimodal Systems. In *Proceedings of the East-West HCI Conference*, 1993. 67

Schäfer, U. Middleware for Creating and Combining Multi-dimensional NLP Markup. In *Proceedings of the Workshop on NLP and XML*, 2006. 178

Scheines, R. and W. Sieg. Computer environments for proof construction. In *Interactive Learning Environments*. 1994. 45, 61

Schiller, M., D. Dietrich, and C. Benzmüller. Proof step analysis for proof tutoring – a learning approach to granularity. *Teaching Mathematics and Computer Science*, 2008. 40, 77, 143, 254

Schmid, H. Unsupervised Learning of Period Disambiguation for Tokenisation. Technical report, IMS, University of Stuttgart, 2000. 146

Schneider, E. Teacher experiences with the use of a CAS in a mathematics classroom. *The International Journal of Computer Algebra In Mathematics Education*, 2000. 61

Schröder, B. and P. Koepke. ProofML – Eine Annotationssprache für natürliche Beweise. *LDV-Forum*, 2003. 53

Schultz, K., E. Bratt, B. Clark, S. Peters, H. Pon-Barry, and P. Treeratpituk. A Scalable, Reusable Spoken Conversational Tutor: SCoT. In *Proceedings of the Workshop on Tutorial Dialogue Systems: With a View toward the Classroom*, 2003. 68

Schwartzman, S. *The Words of Mathematics: An Etymological Dictionary of Mathematical Terms Used in English*, 1994. 116

Searle, J. R. *Speech Acts: An Essay in the Philosophy of Language*, (1969), 1999. 39, 62

Selden, A. and J. Selden. Errors and misconceptions in college level theorem proving. Technical Report 2003–3, Tennessee Technological University, 2003. 29, 60, 134

Self, J., Ed. *Journal of Artificial Intelligence in Education – Special (double) issue on Evaluation*, 1993. 253

Sfard, A. On the Dual Nature of Mathematical Conceptions: Reflections on Processes and Objects as Different Sides of the Same Coin. *Educational Studies in Mathematics*, 1991. 89, 102

Sfard, A. Steering (Dis)Course between Methaphor and Rigour: Using Focal Analysis to Investigate the Emergence of Mathematical Objects. *Journal for Research in Mathematics Education*, 2000. 87

Sfard, A. Learning mathematics as developing a discourse. In *Proceedings of the Conference of North American Chapter of the International Group for the Psychology of Mathematics Education*, 2001. 87, 142

Sgall, P., E. Hajičová, and J. Panevová. *The meaning of the sentence in its semantic and pragmatic aspects*, 1986. 178, 189, 190, 191, 192

Shechtman, N. and L. M. Horowitz. Media inequality in conversation: how people behave differently when interacting with computers and people. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2003. 63

Shuard, H. and A. Rothery, Eds. *Children Reading Mathematics*, 1984. 116

Siekmann, J., C. Benzmüller, A. Fiedler, A. Meier, I. Normann, and M. Pollet. Proof development with Ωmega: The irrationality of $\sqrt{2}$. In *Thirty Five Years of Automating Mathematics*. 2003. 41, 77

Sierpinska, A. *Understanding in Mathematics*, 1994. 29

Silla Jr., C. and C. Kaestner. An Analisys of Sentence Boundary Detection Systems for English and Portugese Documents. In *Proceedings of the CICLing Conference*, 2004. 183

Simon, D. *Checking Number Theory Proofs in Natural Language*. PhD thesis, UT Austin, 1990. 44

Skemp, R. *Intelligence, Learning and Action*, 1979. 88

Skemp, R. *The Psychology of Learning Mathematics*, 1987. 88

Smith, N. A question-answering system for elementary mathematics. Technical Report 227, Stanford University, 1974. 46

Smith, R. and L. Blaine. A generalized system for university mathematics instruction. In *Proceedings of the ACM Symposium on Computer Science and Education*, 1976. 45, 46

Smith, R. and F. Rawson. A Multi-Processing Model for Natural Language Understanding. In *Proceedings of the COLING Conference*, 1976. 46

Smithies, S., K. Novins, and J. Arvo. Equation Entry and Editing via Handwriting and Gesture Recognition. *Behaviour and Information Technology*, 2001. 70

Śniadecki, J. O języku narodowym w matematyce. In Baliński, M., Ed, *Dzieła. Wydanie nowe z 1837*. 1813. 116

Stamerjohanns, H., M. Kohlhase, D. Ginev, C. David, and B. Miller. Transforming Large Collections of Scientific Publications to XML. *Mathematics in Computer Science*, 2010. 182

Steedman, M. *The Syntactic Process*, 2000. 58, 178, 194, 196, 209

Steenrod, N., P. Halmos, M. Schiffer, and J. Dieudonné. *How to write mathematics*, (1975 Reprint), 1973. 31, 60, 86

Stevens, R., P. Write, A. Edwards, and S. Brewster. An audio glance at syntactic structure based on spoken form. In *Proceedings of the Computers Helping People with Special Needs Conference*, 1996. 101

Streeter, L. A. Acoustic determinants of phrase boundary representation. *Journal of the Acoustical Society of America*, 1978. 101

Suppes, P. Computer-assisted instruction at Stanford. In *Computers in the Instructional Process: Report of an International School*. 1974. (1970). 44

Suppes, P., Ed. *University-level computer-assisted instruction at Stanford 1968–1980*, 1981. 44, 45, 61, 253

Suppes, P. and M. Morningstar. *Computer-assisted instruction at Stanford, 1966-1968: Data, Models, and Evaluation of the Arithmetic Programs*, 1972. 61, 253

Suppes, P. and J. Sheehan. CAI course in axiomatic set theory. In *University-level computer-assisted instruction at Stanford 1968–1980*. 1981. 47, 73

Szabolcsi, A. Combinatory Grammar and Projection from the Lexicon. In *Lexical Matters*. 1992. 194

Tall, D. Building theories: The three worlds of mathematics. *For the Learning of Mathematics*, 2004a. 88

Tall, D. Thinking through the three worlds of mathematics. In *Proceedings of the Conference of the International Group for the Psychology of Mathematics Education*, 2004b. 87, 88, 102, 131

Tapia, E. and R. Rojas. Recognition of On-line Handwritten Mathematical Expressions Using a Minimum Spanning Tree Construction and Symbol Dominance. In *Workshop on Graphics Recognition*, 2004. 70, 99

Tesnière, L. *Éléments de syntaxe structurale*, 1959. 189, 190

Thompson, D. and R. Rubenstein. Learning mathematics vocabulary: Potential pitfalls and instructional strategies. *The Mathematics Teacher*, 2000. 27, 100, 101, 116, 117

Tomko, S. and R. Rosenfeld. Shaping spoken input in user-initiative systems. In *Proceedings of the Interspeech Conference*, 2004. 64

Traum, D., J. Bos, R. Cooper, S. Larsson, I. Lewin, C. Matheson, and M. Poessio. A model of dialogue moves and information state revision. Technical report, TRINDI D2.1, 1999. 39

Traum, D. *A Computational Theory of Grounding in Natural Language Conversation*. PhD thesis, University of Rochester, 1994. 39

Trybulec, A. The MIZAR-QC/6000 logic information language. *Association for Literary and Linguistic Computing Bulletin*, 1978. 52

Trzeciak, J. Writing Mathematical Papers in English: A Practical Guide, 1995. 48, 118

Tsovaltzi, D., H. Horacek, and A. Fiedler. Building hint specifications in a NL tutorial system for mathematics. In *Proceedings of the FLAIRS Conference*, 2004. 42, 74

Tsovaltzi, D. *MENON: Automating a Socratic Teaching Model for Mathematical Proofs*. PhD thesis, Universität des Saarlandes, 2010. 42, 74, 254

Tsovaltzi, D. and E. Karagjosova. A View on Dialogue Move Taxonomies for Tutorial Dialogues. In *Proceedings of the SIGdial Meeting*, 2004. 42

Usiskin, Z. Mathematics as a language. In *Communication in Mathematics, K-12 and Beyond*. 1996. 101, 116

van Benthem, J. Categorial Grammar and Type Theory. Technical report, University of Amsterdam, 1987. ILTI Prepublication Series 87-07a. 194

van Eijck, J. and H. Kamp. Representing discourse in context. In *Handbook of Logic & Language*. 1997. 48

Vancoppenolle, J., E. Tabbert, G. Bouma, and M. Stede. A German Grammar for Generation in OpenCCG. In *Proceedings of the GSCL*, 2011. 210

VanLehn, K. Student Modelling. Technical Report PCG-4, Carnegie-Mellon University, 1987. 42

VanLehn, K., C. Lynch, K. Schulze, J. Shapiro, R. Shelby, L. Taylor, D. Treacy, A. Weinstein, and M. Wintersgill. The Andes Physics Tutoring System: Five Years of Evaluations. In *Proceedings of the AIED Conference*, 2005. 62

Verchinine, K., A. Lyaletski, and A. Paskevich. System for Automated Deduction (SAD): a tool for proof verification. In *Proceedings of the CADE Conference*, 2007. 52

Verchinine, K., A. Lyaletski, A. Paskevich, and A. Anisimov. On correctness of mathematical texts from a logical and practical point of view. In *Proceedings of the MKM Conference*, 2008. 41

Vershinin, K. and A. Paskevich. ForTheL – the language of formal theories. *International Journal of Information Theories and Applications*, 2000. 41, 52

Vierhuff, T., B. Hildebrandt, and H.-J. Eikmeyer. Effiziente Verarbeitung deutscher Konstituentenstellung mit der Combinatorial Categorial Grammar. *Linguistische Berichte*, 2003. 209

Vo, B., C. Benzmüller, and S. Autexier. Assertion application in theorem proving and proof planning. In *Proceedings of IJCAI*, 2003. 41

Vuong, B., Y. He, and S. Hui. Towards a web-based progressive handwriting recognition for mathematical problem solving. *Expert Systems with Applications*, 2010. 70

Wagner, M. and H. Lesourd. Using TEX macs in Math Education: An exploratory Study. In *Proceedings of the MathUI Workshop*, 2008. 43, 168

Wagner, M., S. Autexier, and C. Benzmüller. PlatOmega: A Mediator between Text-Editors and Proof Assistance Systems. In *Electronic Notes in Theoretical Computer Science*, 2007. 41, 43

Walker, D., D. Clements, M. Darwin, and J. Amtrup. Sentence Boundary Detection: A Comparison of Paradigms for Improving MT Quality. In *Proceedings of MT Summit VIII*, 2001. 146

Walker, M. and R. Passonneau. DATE: A Dialogue Act Tagging Scheme for Evaluation of Spoken Dialogue Systems. In *Proceedings of the Human Language Technology Conference*, 2001. 39

Ward, A. and D. Litman. Cohesion and learning in a tutorial spoken dialog system. In *Proceedings of the FLAIRS Conference*, 2006. 76

Wells, C. *A Handbook of Mathematical Discourse*, 2003. 86, 103, 138

Wells, C. Abstract Math website, n.d. `http://abstractmath.org` 86

Wells, M. MADCAP: a scientific compiler for a displayed formula textbook language. *Communications of the ACM*, 1961. 97

Wenzel, M. Isabelle/Isar – a Generic Framework for Human-Readable Proof Documents. In *From Insight to Proof: Festschrift in Honour of Andrzej Trybulec*. 2007. 52

Wigmore, A., E. Pflügel, G. Hunter, J. Denholm-Price, and M. Colbert. TalkMaths Better! Evaluating and Improving an Intelligent Interface for Creating and Editing Mathematical Text. In *Proceedings of the Conference on Intelligent Environments*, 2010. 69

Wöllstein-Leisten, A., A. Heilmann, P. Stepan, and S. Vikner. *Deutsche Satzstruktur. Grundlagen der syntaktischen Analyse*, 1997. 207

Wolska, M. A language engineering architecture for processing informal mathematical discourse. In *Proceedings of the Workshop ''Towards a Digital Mathematics Library''*, 2008. 32

Wolska, M. The Language of Learner Proof Discourse: A Corpus Study on the Variety of Linguistic Forms. In *Proceedings of the Workshop on Computational Models of Natural Argument*, 2012. 32, 149

Wolska, M. and M. Buckley. A Taxonomy of Task-related Dialogue Actions: The Cases of Tutorial and Collaborative Planning Dialogue. In *Text Resources and Lexical Knowledge*. 2008. 42, 151, 154, 256

Wolska, M. and M. Grigore. Symbol declarations in mathematics. In *Proceedings of the Workshop ''Towards a Digital Mathematics Library''*, 2010. 280

Wolska, M. and I. Kruijff-Korbayová. Analysis of Mixed Natural and Symbolic Language Input in Mathematical Dialogs. In *Proceedings of the ACL Conference*, 2004a. 32, 47, 145, 175, 177, 205

Wolska, M. and I. Kruijff-Korbayová. Building a dependency-based grammar for parsing informal mathematical discourse. In *Proceedings of the Text, Speech and Dialogue Conference*, 2004b. 32

Wolska, M. and I. Kruijff-Korbayová. Factors influencing input styles in tutoring systems: the case of the study-material presentation format. In *Proceedings of the Workshop on Language-enhanced Educational Technology*, 2006a. 32, 61, 149, 162

Wolska, M. and I. Kruijff-Korbayová. Modeling anaphora in informal mathematical dialogue. In *Proceedings of the SemDial Workshop*, 2006b. 32, 205

Wolska, M., I. Kruijff-Korbayová, and H. Horacek. Lexical-semantic interpretation of language input in mathematical dialogs. In *Proceedings of the Workshop on Text Meaning and Interpretation*, 2004a. 32, 205

Wolska, M., B. Vo, D. Tsovaltzi, I. Kruijff-Korbayová, E. Karagjosova, H. Horacek, M. Gabsdil, A. Fiedler, and C. Benzmüller. An annotated corpus of tutorial dialogs on mathematical theorem proving. In *Proceedings of LREC*, 2004b. 32, 61

Wolska, M., M. Buckley, H. Horacek, I. Kruijff-Korbayová, and M. Pinkal. Linguistic Processing in a Mathematics Tutoring System: Cooperative Input Interpretation and Dialogue Modelling. In *Resource-Adaptive Cognitive Processes*. 2010. 32, 175

Wolska, M., M. Grigore, and M. Kohlhase. Using discourse context to interpret object-denoting mathematical expressions. In *Proceedings of the Workshop ''Towards a Digital Mathematics Library''*, 2011. 280

Wray, A. and M. Perkins. The functions of formulaic language: an integrated model. *Language & Communication*, 2000. 118

Yankelovich, N., G. Levow, and M. Marx. Designing SpeechActs: Issues in speech user interfaces. In *Proceedings of the Conference on Human Factors in Computing Systems*, 1995. 63

Youmans, G. Measuring Lexical Style and Competence: The Type-Token Vocabulary Curve. *Style*, 1990. 170

Young, S. Probabilistic methods in spoken-dialogue systems. *Philosophical Transactions of the Royal Society of London*, 2000. 39

Zazkis, R. and C. Gunn. Sets, Subsets, and the Empty Set: Students' Constructions and Mathematical Conventions. *Journal of Computers in Mathematics and Science Teaching*, 1997. 110

Zhang, L. and R. Fateman. Survey of User Input Models for Mathematical Recognition: Keyboards, Mice, Tables, Voice. Technical report, University of California, 2003. 70

Zinn, C. *Understanding Informal Mathematical Discourse*. PhD thesis, Universität Erlangen-Nürnberg, 2004. 44, 48, 56, 85, 120, 145, 149, 177, 186

Zinn, C. Supporting the formal verification of mathematical texts. *Journal of Applied Logic*, 2006. 35, 48, 176

Zinn, C., J. Moore, and M. Core. A 3-Tier Planning Architecture for Managing Tutorial Dialogue. In *Proceedings of the ITS Conference*, 2002. 68

Zinn, C., J. Moore, M. Core, S. Varges, and K. Porayska-Pomsta. The BE&E Tutorial Learning Environment (BEETLE). In *Proceedings of the SemDial Workshop*, 2003. 42

Zoltan-Ford, E. How to get people to say and type what computers can understand. *International Journal of Man-Machine Studies*, 1991. 64

Truth and proof are central to mathematics. Proving (or disproving) seemingly simple statements often turns out to be one of the hardest mathematical tasks. Yet, doing proofs is rarely taught in the classroom. Studies on cognitive difficulties in learning to do proofs have shown that pupils and students not only often do not understand or cannot apply basic formal reasoning techniques and do not know how to use formal mathematical language, but, at a far more fundamental level, they also do not understand what it means to prove a statement or even do not see the purpose of proof at all. Since insight into the importance of proof and doing proofs as such cannot be learnt other than by practice, learning support through individualised tutoring is in demand.

This volume presents a part of an interdisciplinary project, set at the intersection of pedagogical science, artificial intelligence, and (computational) linguistics, which investigated issues involved in provisioning computer-based tutoring of mathematical proofs through dialogue in natural language. The ultimate goal in this context, addressing the above-mentioned need for learning support, is to build intelligent automated tutoring systems for mathematical proofs. The research presented here has been focused on the language that students use while interacting with such a system: its linguistic properties and computational modelling. Contribution is made at three levels: first, an analysis of language phenomena found in students' input to a (simulated) proof tutoring system is conducted and the variety of students' verbalisations is quantitatively assessed, second, a general computational processing strategy for informal mathematical language and methods of modelling prominent language phenomena are proposed, and third, the prospects for natural language as an input modality for proof tutoring systems is evaluated based on collected corpora.

ISBN 978-3-86223-111-9