Diplomarbeit

Mechanizing Proof Step Evaluation for Mathematics Tutoring - The Case of Granularity

Marvin R. G. Schiller

Fachrichtung Informatik
Universität des Saarlandes
D-66041 Saarbrücken, Germany
schiller@ags.uni-sb.de

Mentoring and topic by
Dr. Christoph Benzmüller
Fachrichtung Informatik
Universität des Saarlandes
D-66041 Saarbrücken, Germany
chris@ags.uni-sb.de

November 20, 2005

Hiermit versichere ich an Eides statt, dass ich diese Arbeit selbständig verfasst habe und keine anderen als die angegebenen Hilfsmittel verwandt habe.

Saarbrücken, November 20, 2005

Abstract

The motif of this thesis is the application of deduction techniques to tutorial dialogs in mathematics. The aim is to characterize different degrees of “granularity” of proof steps, which designates their argumentative size. This aspect of mathematical proof steps, also referred to as grain size, is still under-investigated in the theorem proving community. However, the ability to reflect on the granularity of proof steps is important for teaching mathematics. We propose an approach towards the automatization of the granularity assessment for a given proof fragment, and perform a first analysis on an empirically collected sample of mathematical dialogs from a Wizard-of-Oz study that is also part of this thesis. We show how such an analysis is performed with a first manifest hypothesis on how a granularity measure is obtained. This hypothesis relates the granularity level of a mathematical statement to the number of inference steps required for its justification, which is tested for justifications in two different natural deduction calculi. The outcome does not provide sufficient evidence for this first hypothesis. However, it clarifies its deficiencies.

Acknowledgments

At this point I would like to thank my mentor Christoph Benzmüller for the topic of this thesis and his valuable counseling. I am particularly grateful for the excellent working environment provided by Prof. Jörg Siekmann, who was a source of enthusiasm and inspiration for me. I am also indebted to the members of the Dialog project and the Ωmega group, who provided an inspiring working atmosphere. My special thanks go to Bao Quoc Vo, who provided additional mentoring for me at the beginning of my work, Martin Pollet, who initiated me to the Ωmega system, Henri Lesourd, who introduced me to the inner life of TeXmacs, and both Magdalena Wolska and Helmut Horacek, who were my main collaborators in the recent empirical Dialog study and with whom this challenging task became very enjoyable. A special mention goes to Chad E. Brown and Claus-Peter Wirth, who provided me with guidance and insight into many topics in logic, and to Dimitra Tsovaltzi, Dieter Hutter, Serge Autexier and Armin Fiedler for valuable discussions. I would like to thank my fellow students Mark Buckley, Dominik Dietrich, Marc Wagner and Jürgen Zimmer, who have helped me in many ways, including discussions and proofreading.

Bei meinen Eltern bedanke ich mich herzlich für ihre moralische und finanzielle Unterstützung.

Contents

Chapter 1

Introduction

Interactive and automated reasoning techniques in mathematics have reached a considerable degree of maturity and practical relevance (consider for example the automated proof of the four-color theorem [3], [42]). However, the interaction with such systems generally requires practice and skills a mathematician might not need to acquire if working with pen and paper. The success of computer-assisted mathematics depends on the naturalness of interaction between the user and the system.

This is particularly true for mathematical tutoring systems. The DIALOG project group [6] at Saarland University investigates the development of techniques for dialog systems in mathematics. These include domain reasoning techniques from theorem proving for the analysis of dialog contributions with mathematical content. In the scope of the DIALOG project, empirical studies have been carried out which contribute to the modeling of a dialog system and the partial implementation of a demonstrator. Recent investigations [9] in the DIALOG project identify the need for analyzing the *granularity* of proof steps proposed by the user. Here, granularity designates the size of a proof step w.r.t. its argumentative complexity. One postulate of this thesis is that proofs at different granularity levels can be characterized by aspects like abstraction, linguistic representation, and cognitive effort involved in finding or verifying their proof steps.

The motif of this thesis is the question whether an automated procedure will be able to provide a granularity measure for a given proof. Therefore, a method for the systematic granularity analysis of proofs is developed and tested with a corpus of mathematical dialogs from an empirical study. This exploratory study reflects in its design the particular aim to observe granularity phenomena and provides a large corpus of dialogs on mathematical proofs. The hypothesis is investigated that granularity judgments of human reasoners w.r.t a mathematical statement are related to the number of proof steps such a statement requires as a justification. Therefore, justifications are systematically reconstructed for samples from the corpus. This is done with two different natural deduction calculi (Gentzen's calculus NK [20] and Rips's PSYCOP proof procedure [40]) within a framework for theorem proving, the ΩMEGA system [23]. The results are then compared to the empirical data under the aspect of granularity.

Unlike the aspect of correctness of proof steps, the aspect of granularity has been less investigated by the theorem proving community so far. Another difference between correctness and granularity is that a verdict on correctness has an absolute quality (a theorem can be true, false or have an unknown status), whereas granularity judgments by mathematicians range over a subjectively experienced continuum. The topic of this thesis is situated at the intersection of the sub-field of deduction in artificial intelligence with the area of deductive reasoning in cognitive psychology and the field of mathematics tutoring.

1.1 Contributions

This work provides - besides a tempted definition of the notion of "granularity" - the identification of different aspects of conceptual, linguistic and cognitive nature that determine a "granularity level" of a mathematical proof argument. As a part of the thesis, a complex empirical study has been designed, which involved programming of a highly sophisticated environment for Wizard-of-Oz studies in mathematics and engineering. The author also played a vital part in the preparation and practical realization of the experiment, which involved up to four experimental sessions per day in a four-week period. The study provides a corpus of 37 dialogs in natural language and formulae together with explicit judgments by the tutors about the correctness, grain size and relevance of the student's utterances. Being a promising theory for a cognitively plausible calculus, all of the proof rules in the PSYCOP theory for propositional logic, and a substantial part of the complex rules for first-order logic have been implemented in the ΩMEGA framework. This allows the modeling of proofs from the empirical corpus of mathematical problems according to the PSYCOP theory. Like this, PSYCOP proofs were created that are more complex than the examples given in Lance Rips' influential work "The Psychology of Proof"[40], and that reflect genuine mathematical problems rather than toy examples. This work also comprises design and implementation of a freely configurable framework for the mechanized granularity analysis of proofs in arbitrary calculi. It was used in the first analysis of samples from the corpus, which also forms a part of this thesis. The proofs for these mathematical statements in the PSYCOP theory and Gentzen's natural deduction calculus [20] were compared under the hypothesis that the number of proof steps identifies the granularity level of the statements.

1.2 Significance of this Work

In different sub-areas of artificial intelligence, the need to investigate the role of granularity has been identified (e.g. [24], [35]). However, theories and methods dealing with granularity are still limited. The approach presented in this thesis can thus be understood as pioneering work in this direction.

Chapter 2

Granularity of Proofs

The impetus for this work was provided by the research in the ΩMEGA group at Saarland University and the DIALOG project in the Collaborative Research Center *Resource Adaptive Cognitive Processes* [1].

The DIALOG project is concerned with the development of natural language dialog techniques for the domain of mathematics. This includes techniques for the analysis of natural language, dialog management, the application of mathematical domain reasoning and mathematics tutoring.

In the overall architecture of the conceptualized dialog system, the work presented in this thesis is situated on the side of the mathematical domain reasoning, which supports dialog management with domain-specific information. Especially for building an intelligent tutoring system (ITS) for mathematics, a thorough analysis of the mathematical content in a dialog is crucial.

The ΩMEGA group [23] is concerned with the development of intelligent systems to support the work of mathematicians and engineers. The ΩMEGA system is a mathematical assistant system that provides a platform for interactive and automated theorem proving.

In this context, it is possible to feed mathematical statements from the input of a dialog system into a proof checker for determining their correctness. However, a Wizard-of-Oz study in the DIALOG project in 2003 [5] showed that in the domain of tutoring proofs, also other qualities of mathematical proofs are important to analyze, namely the granularity of proof steps (also termed "grain size") and relevance.

In [5] and [9], the challenging task of developing an interactive proof environment is discussed with respect to the empirical study of tutorial dialogs in mathematics. The idea presented in [5] is to evaluate proof steps w.r.t. three aspects; accuracy (correctness), relevance and granularity. The characterization of granularity states that: "The level of granularity reflects the number of proof steps required to establish the asserted proof step." (cf. [5]).

This measure of granularity depends on the choice of a particular proof system. Therefore, the accuracy of this method will depend on how well the number of proof steps in the employed calculus fits the proof sizes as they subjectively appear to human reasoners. This is an empirical matter, which motivates the recent empirical study to be presented in Chapter 3.

On the other hand, there is evidence[1] that the perceived size of a mathematical argument - or its complexity - does not only depend on the number of justifying steps in a particular calculus. That point of view ignores that the performance of human reasoners is determined by cognitive processes that do not only deal with the formal content of a mathematical problem, as it is usually presented to a theorem prover. Human reasoners also engage in perception, language processing and possibly perform shifts in the problem representation, for example. This motivates the study on cognitive theories on reasoning, such as the "Psychology of Proof" theory by Lance Rips presented in Chapter 7. It also motivates the study of those factors that contribute to the argumentative complexity of proof steps in more detail.

The following section presents a first attempt to characterize the dimensions of granularity in mathematical proofs. In the later chapters, the aim to develop techniques to mechanically analyze the granularity of mathematical arguments is pursued. Such a mechanized component will potentially provide the other modules in a mathematical assistant or dialog system with specific information on the granularity dimensions of proofs.

2.1 What is the Granularity of a Proof?

We postulate that the notion of *granularity* of a proof can be characterized by the following, not necessarily exhaustive list of aspects:

- Abstraction: Specification of a mathematical statement or problem on a particular level of detail.
- Explicitness: The ratio between openly communicated information versus hidden information in a mathematical statement.
- Cognitive Effort: The amount of cognitive resources needed by the human reasoner to produce or to understand a mathematical statement.

We will discuss each of these aspects in turn, and show how the amalgam of these aspects forms the notion of granularity. Even though these three aspects are

[1] The corpus collected in the empirical study described in Chapter 3 and presented in Chapter 5 contributes towards that evidence.

considered separately in the following sections, it is not unusual to find interactions between them.

2.2 Abstraction

Abstraction is one of the basic principles of mathematics. For example, a significant step towards modern mathematics was the introduction of *variables*, which are abstract placeholders representing particular objects. The merit of introducing Latin symbols as variables into mathematics is attributed to François Viète (1540-1603).

Abstraction can be defined formally as a process of mapping the representation of a problem (the ground representation) onto a new representation (the abstract representation). Usually, a representation is considered *more abstract* than another if it contains less details. Which properties of the problem are preserved in the process of abstraction, and which properties are omitted, relates to the question which properties are *relevant* for the purposes of the problem representation.

In the study of formal systems, the authors of [21] formally define abstraction as follows: A pair of formal systems Σ_1, Σ_2, in which statements are represented in the languages Λ_1 and Λ_2, respectively, together with a mapping (an effective total function) $f : \Lambda_1 \rightarrow \Lambda_2$ is an abstraction. That is, the abstraction process maps statements from one theory onto the other, and vice versa.

In mathematical proofs, one can find (at least) two major classes of objects that are subject to abstraction:

- mathematical objects, e.g. sets,
- mathematical arguments, e.g. inference steps, methods, justifications etc.

As an example for abstraction on mathematical objects, consider abstract characterizations of an "equivalence relation". An equivalence relation is characterized by its distinguishing properties (it is a relation that is reflexive, symmetric and transitive), independent of other details of a concrete relation. But an equivalence relation is also a set. Different viewpoints can be taken on such an equivalence relation, where different properties become relevant, whereas others are moved to the background. For example, to determine whether two equivalence relations S and R are equal, it suffices to consider them as sets and determine if all elements in S are in also in R and vice versa. To determine whether the intersection $R \cap S$ is again an equivalence relation (which is indeed the case), a more detailed view is required, which takes into account the properties of reflexivity, symmetry and transitivity.

Assume R is an asymmetric relation on a set M. If R is not empty (i.e. $R \neq \emptyset$), then it holds:

$$R \neq R^{-1}.$$

Figure 2.1: Example Problem

But mathematical arguments can be viewed on different levels of detail, too. For example, the little theorem in Figure 2.1 can be justified with "proof by contradiction". But it can also be justified by a more detailed justification, for example:

"Assume that $R = R^{-1}$. Then there exists $r \in R$ such that $r \in R^{-1}$. Since R is asymmetric, $r \notin R^{-1}$. This leads to a contradiction."

The different degrees of mathematical abstraction can be expressed in varying degrees of explicitness. This has an effect on the linguistic representation of proofs, which is discussed in the following section.

2.3 Explicitness

Conveying a proof in written or spoken form is a communication act. It can be characterized by the information content of the speech acts. In mathematical argumentation, it is common to omit certain information - this is referred to as *underspecification*. An investigation on underspecification phenomena in mathematical dialogs is presented in [9].

Explicitness refers to the ratio between explicitly given relevant information in a mathematical explanation versus hidden, i.e. implicit but relevant information that the recipient of the explanation has to construct himself. Why are mathematical arguments left underspecified?

Very often, the author and the recipient share a common ground. That is, they have similar mathematical background knowledge, which may allow the recipient to reconstruct those mathematical facts that are left implicit. An underspecified description of a proof step leaves the recipient with a "search space" of possible concrete instances of the argument in question. A higher degree of explicitness reduces the uncertainty of the recipient (thus reducing the "search space", if the problem is conceived as a search problem).

On the other hand, a very detailed description of a mathematical argument can lead to an unnecessarily long textual representation. The recipient may not need all that information. In the worst case, she or he will miss the point of the explanation. Thus a very detailed explanation can violate the communication goal, too. Like other forms of communication, mathematical explanations have to fulfill the Gricean maxims, which comprise conciseness, among others.

Knowing the audience can thus make the communication more efficient. This contributes to the diversity of styles of mathematical explanations. Consider - for example - the differences between a mathematical textbook and a conference paper.

2.4 Cognitive Effort

Human mathematicians often classify proofs as being "easy" or "hard", depending on the amount of knowledge, insight and work they have to spend on finding it, or, once it is found, in understanding it.

Understanding a proof is a highly demanding cognitive act. It involves perception (visual or auditory), language processing and memory. Each of these factors has an influence on the performance of a human mathematician on a task. The cognitive resources of a human mathematician are limited. For example, the typical capacity of the working memory is about 5-7 memory items[2]. One of the pioneer works in establishing the working memory span is [36]. The effective use of these resources distinguishes an expert mathematician from a beginner. But the limitations also require mathematicians not to over-strain a potential dialog partner or reader. In mathematics education, this overstraining usually leads to discouragement and irritation. The other extreme of under-challenging the conversational partner can also have an irritating effect and produce boredom.

In conversations (not only those that deal with proofs), human reasoners like to be given "problems" that are both stimulating and challenging, but still tractable (given their resource limitations). Thus, the cognitive effort to solve these little problems constitutes a dimensions of *granularity* of mathematical proofs.

The dimension of cognitive effort also affects the dimensions of abstraction and explicitness. Abstraction is a means to reduce the cognitive effort of an otherwise unmanageable task. The linguistic representation of mathematical facts is often adapted to the cognitive capacities assumed in the reader or conversational partner, too.

Determining these factors requires psychological investigation into the functioning of the human mind. The relevant processes may heavily depend on differences between individuals. The cognitive effort a proof requires is thus no absolute quantity.

Nevertheless, some general properties of cognitive processes seem to evolve from investigations in cognitive psychology, and cognitive architectures (like SOAR, EPIC or ACT-R) can simulate human subjects on isolated tasks to establish statistically significant correspondences. Cognitive theories on mathematical problem solving have been simulated in ACT-R, for example in the domain of early algebra

[2]These items can be simple or complex items, depending on training. Mnemonic techniques make use of this fact.

problem solving [32]. The aim of ACT-R is to provide one unified architecture to explain different cognitive activities, not only in mathematics, but also language processing or learning, for example.

2.5 Related Theories

This section introduces two works that deal with granularity, and which are both situated in the wider context of artificial intelligence. There are also theories on the nature of human logical reasoning developed by psychologists. Examples for these are the PSYCOP theory by Lance Rips [40] and "Mental Model" theory by Johnson-Laird [29]. Because the PSYCOP theory plays a central role in this thesis, Chapter 7 is fully dedicated to it. The topic of this thesis is situated in the intersection between the field of deduction in artificial intelligence and the psychology of reasoning.

2.5.1 Jerry R. Hobbs' "Granularity"

In Hobbs' conference paper [24], a framework for a theory of granularity is developed. The author proposes a simplification process in which a complex theory is mapped onto a simpler (more coarse-grained) theory via an indistinguishability relation $\sim$. This indistinguishability relation $\sim$ is defined on the domain of interpretation S_0 of a logical theory T_0 as follows:

$$\forall x \; \forall y \; (x \sim y \Leftrightarrow (\forall p \in R \; (p(x) \Leftrightarrow p(y)))).$$

R is a set of "relevant" predicates. That is, x and y are indistinguishable, if no relevant predicate distinguishes between them. The problem of determining which predicates are relevant in a given situation is seen as a "very hard problem" (cf. [24], p.2). A mapping κ can be defined taking each element of S_0 to the set of equivalence classes of S_0 with respect to $\sim$. Thus, a complex theory can be collapsed into a simpler theory.
The translation in the other direction, from a coarse-grained level of granularity to a fine-grained level, is termed "articulation" in [24].

Discussion The approach taken in [24] presents abstraction (and articulation) as a method to translate between different levels of "granularity" of logical theories. This is done in the hope to solve the "representation problem" in artificial intelligence. In this case, finding the right granularity of a problem representation means simplification in the spirit of abstraction, making the problem computationally more tractable but preserving the essential properties. Hobbs' definition of the indistinguishability relation bears a striking resemblance with the well-known definition of Leibniz equality (even though Hobbs himself does not mention this in [24]). The only difference is that Leibniz equality stipulates that two objects are "equal" if they have the same properties, whereas Hobbs' similarity relation takes into account a "relevant" subset of these properties. The main work - determining which properties are relevant - thus remains. Hobbs' work emphasizes the role of abstraction in the theory of granularity. His work does not specifically deal with linguistic or cognitive aspects of logics, but with a general approach to problem solving in artificial intelligence.

2.5.2 "Granularity Hierarchies"

In [35] a framework for the representation of granularity hierarchies is presented, which is based on semantic networks. It is applied for diagnosis in an intelligent tutoring environment in the domain of LISP programming. With this, a granularity-based system that recognizes LISP programming strategies has been implemented as a response analyst for the SCENT (Student Computing ENvironmenT) architecture, an "intelligent adviser" for novice LISP programmers.

The authors emphasize the importance of studying the notion of granularity, but also that "the explicit study of granularity in AI has essentially been overlooked." They give two reasons for their interest in granularity, namely pedagogical reasons and robustness of diagnosis. The pedagogical aspect concerns the necessity of an appropriate level of instruction, with only smooth shifts in grain size. The diagnosis aspect concerns the ability to understand a student 's behavior at various levels of detail.

Two orthogonal dimensions of granularity are identified, *abstraction* and *aggregation*, which "somewhat" (cf. [35]) correspond to the concepts of "isa" and "part-of" relations in semantic networks. The representation for granularity consists of a network of nodes, with two types of directed links between these nodes that correspond to either abstraction or aggregation. Some objects may be maximal aggregations, i.e. objects that are not part of other objects.
A principal abstraction hierarchy is obtained by only considering the maximal aggregations with links corresponding to the abstraction relation, thus forming a uniquely-rooted acyclic graph. On the other hand, each maximal aggregation object is the root of another directed acyclic graph linked by the aggregation relation.

After having constructed the "granularity hierarchy", the authors compare their approach to Hobbs's. The indistinguishability relation is implicitly represented by the structure of the granularity hierarchy. Indistinguishability between two objects is defined relative to another object, such that two descendants of an object (in either of both relations) are indistinguishable with respect to the "ancestor" object. Simplification and articulation in the sense of Hobbs are achieved by shifts in the graph.

In the LISP tutoring environment, observations from the student's programming attempt are collected by so-called "K-clusters", which characterize a particular perspective on the object under consideration. In the simplification process, the shift towards a coarser grain size is achieved by a simplification operator which combines these observations. Thus, the more "coarse" object is more than just the sum of its parts, since the simplification operator makes sure that the combined parts fit together appropriately.

This network only captures relative granularities between objects, no absolute quantities. In particular, "the degree of precision of a measurement in the real world" and the granularity of the corresponding representation " have no relationship to one another" ([35], page 10). The authors assert "granularity is relative, not absolute".

Discussion This approach is based on the observation that granularity comprises orthogonal dimensions. In its generality, this approach could also be applied to other real-world classification problems.

The granularity hierarchies are pre-specified for the given tutoring domain. What is gained is the flexibility to identify only partially recognizable objects at different levels. This is somewhere half-way between a "golden solutions" approach (i.e. the system compares proposed solutions to a corpus of pre-specified solutions) and fully dynamic evaluation of the observed behavior. In the domain of mathematical proof there is the potential for the latter, since often - in principle - mathematical arguments can be checked dynamically.

2.5.3 Cognitive Principles in the Design of Computer Tutors

From their work in building tutoring systems with the help of the cognitive architecture ACT* (the predecessor theory of ACT-R), the authors of [2] derive eight principles for the design of computer tutors. They advocate that the pedagogical design of such a computer tutor must be based on a detailed cognitive model on how students solve problems and learn. These processes are simulated in the ACT* theory. There, the cognitive architecture is modeled as a production system, with production rules (i.e. condition-action pairs) that describe individual steps of cognition. Learning is simulated by a mechanism that systematically combines different production rules to form new production rules.

The principles that guide the development of the tutoring systems, a geometry tutor and a LISP tutor, are obtained from a series of observations of high school and college students while they were learning and solving relevant problems.

One of the principles, namely the seventh, states the necessity to "adjust the grain size of instruction with learning". The authors state that the productions in the ACT* theory define the grain size with which a problem is solved. They observe that "beginning students need to be led through some inferences in mini-steps, while advanced students prefer to plan in multi-step inferences" (cf. [2], p.118). This was not yet achieved by the geometry tutor described in [2], where grain size always corresponded to a single step of inference.

Discussion The work presented in [2] makes the appropriate level of grain size one of the principles for building tutoring systems. This refers to the grain size of instruction, but the authors also stress the importance of a detailed model of the student's problem solving process. There, the production rules in the cognitive architecture become the normative unit for granularity.

2.6 Analysis of Granularity

The work presented in the following chapters is concerned with the application of theorem proving techniques to the analysis of granularity in mathematical proofs. As a basis for the investigations serves a Wizard-of-Oz study presented in the next chapter, which yields a corpus of tutorial dialogs in mathematics. The idea is to reconstruct the given proof steps in a suitable calculus. In the case of this thesis, Gentzen's Natural Deduction [20] and Rips's PSYCOP [40] were considered as candidates. The reconstructions are then evaluated in a systematic way in an analysis framework to obtain an indicator of their "grain size". The framework and a first analysis are presented in Chapters 6 and 8. This kind of analysis supports the evaluation of those factors that are linked to abstraction and cognitive effort as discussed above, but it will be insensitive to linguistic factors. So in the remaining chapters of this thesis, linguistic aspects will be factored out. This does not mean that they play a minor role with respect to the other qualitative aspects of proofs, however. They require additional techniques from computational linguistics that are addressed by the DIALOG project group, but which are beyond the scope of this thesis.

Chapter 3

A Wizard-of-Oz Study on Tutoring Mathematical Proofs

In the multidisciplinary DIALOG project [6] in the Collaborative Research Center SFB 378 *Resource-adaptive Cognitive Processes* [1], we conducted an exploratory study on the tutoring of mathematical proof in summer 2005. This study was a follow-up on a first Wizard-of-Oz Study in 2003. This first study, which is presented in [7], has provided the impetus for dedicating this thesis to the aspect of granularity in mathematical proofs. Therefore, the phenomenon of granularity was one particular focus in the new study in 2005. Furthermore, a new mathematical domain was investigated with respect to the former experiment. Whereas the mathematical domain of the previous experiment was sets, operations on sets and their power-sets, now the domain of relations was investigated, with an emphasis on advanced concepts like the inverse of a relation and relation composition.

3.1 Aims

The main aim of this Wizard-of-Oz study was the collection of empirical data to study tutorial dialog on proofs. The study of the effects of granularity was one major interest. The design of the exercises and the material was driven by this aspect, and the tutors had to make corresponding judgments for each discourse contribution by the student.

A second aspect of the experiment was the study of natural language and dialog in the context of tutoring mathematical proofs, in particular:

- Linguistic phenomena; like the mixing of mathematical formulae and natural language (sometimes termed "mathural language", as in [38]),
- underspecification phenomena,

Figure 3.1: View of the subject room

- mutual interleaving and dependencies between natural language analysis and non-trivial domain reasoning,
- assessment of utterances under tutorial aspects.

3.2 Experimental Design

"Wizard-of-Oz" is a rapid-prototyping method for systems costly to build or requiring new technology (cf. [34]). A human "wizard" simulates the system's intelligence and interacts with the user through a real or mock computer interface.

The main idea for the experiment was to give the subjects a set of concrete proof tasks. They were instructed to collaboratively and interactively solve these exercises with the help of a dialog tutoring system. This "tutoring system" consisted of a human tutor, the "wizard", and the DiaWOz-II software, a software that mediates between both the wizard and the participant. The communication consisted

Figure 3.2: View of the wizard's room

only of typed language (words and formulae). Speech and gesture were recorded, but not part of the communication transmitted between the student and the tutor via DiaWOz-II .

Two adjacent rooms were used for the experiment, both with one door each facing to the corridor. The subject was placed in one of these rooms with a PC running the "student" interface of the DiaWOz-II software, as shown in Figure 3.1. The wizard, and - during the course of the experiment - also the experimenter were placed in the second room, together with the server for the DiaWOz-II tool and most of the recording facilities. This room is shown in Figure 3.2.

The software for the student was presented on a windows PC with a Pentium 4 processor and 1 GB of RAM with a 17 inch TFT screen. The actual client processes for the Wizard-of-Oz tool ran on the server PC, and were displayed on the student's computer via an SSH session. The wizard communicated with the student via a second Wizard-of-Oz tool client process, running on a second computer connected to the server PC.

3.2.1 Participants

The subjects in our study were 37 university students aged between 20 and 34 years (average age: 23.6) who received monetary compensation for their participation. They had to fulfill the requirements of being native German speakers, and of having at least visited one university-level mathematics course. The participants have different study backgrounds, such as computer science, computational linguistics, bioinformatics, mathematics, economics, intercultural communication and material science (engineering).

Among the 37 subjects, 27 had taken mathematics as "specialized course" in the scope of their a-levels ("Abitur"), a strong indicator of a fair mathematical background w.r.t. the basics of mathematics. In the pre-test questionnaire all 37 subjects indicated that they are familiar with the concepts of set, set union and intersection. 28 subjects (=75,5 %) indicated familiarity with the notion of a "pair", 31 (=83,8 %) subjects the familiarity with the concept of a "relation", 19 (=51,3 %) with the "relation inverse" and 20 (54,1 %) with "relation composition".

Among the subjects there was one participant with Serbo-Croatian as her mother-tongue, but living in Germany since 13 years. Therefore, she was treated separately in the evaluations of the experiment.

The role of the "dialog tutoring system" was taken over by experts in the field of mathematics, the so-called "wizards". These four wizards all hold a degree in mathematics. One of them is the lecturer of the university courses "Mathematics Foundations of Computational Linguistics I and II" and "Mathematical Logic". Another wizard started a teaching career at a high school during the time he participated in the experiment. The third wizard has work experience as a tutor for linear algebra and computer literacy. At the time of the experiment he applied as a mathematics teacher at a school. The fourth wizard has experience as a mathematics tutor at university.

All these wizards were chosen such that they did not have knowledge on automated theorem proving, since this knowledge might have had an influence their mathematical style.

The experimenter role was taken in turn by three members of the Dialog project group. Wizard and experimenter were assisted by a third person in the "wizard's room" who was also controlling the technical setup. This was one out of two researchers in the Dialog team who had been involved in the design of the experiment and the programming of the DiaWOz-II software.

3.2.2 Data Recording

The behavior of the subjects as well as her or his interaction with the experimenter and the hypothetical system was recorded during the entire experimental session.

The dialog between the subject and the hypothetical tutoring system took place in written form, i.e. the subject typed a mix of text and formulae into the interface provided by the DiaWOz-II software described in Chapter 4. This dialog consisting of contributions by the subject, the answers from the wizard and automated messages from the system was automatically written into a log-file by the DiaWOz-II software. Each entry in the log-file contains a time stamp identifying the clock time the corresponding message was sent.

Furthermore, video recordings of the subject's computer screen were made (at a resolution of 800×600 pixels), which show the use of the interface by the subject in detail. The upper body of the subject was filmed with a dome camera that was mounted below the ceiling of the room, which allowed to observe and record body movements and facial expression. These recordings are in the *avi* (audio video interleave) format with a resolution of 720×576. Unfortunately, some of the video recordings were truncated or even lost due to hardware and software problems.

Audio recordings were made for the subject, the experimenter and the wizard. The subject was instructed to "think aloud" and to speak comments into a microphone on the desk. The wizard was equipped with a head microphone and encouraged to make verbal comments about his decisions for later reference. The audio recordings consist of two separate audio tracks for subject and wizard, and a third track that furthermore includes the communication between the subject and the experimenter. The audio recordings were stored digitally on the hard-disk of a dedicated computer.

Before each experimental run, the experimenter indicated to the subjects that their tutoring sessions were recorded.

3.2.3 Usability Evaluation

Subjects were given a pre-and a post test questionnaire to be filled out on the same computer where the dialog interface was run. This served three goals:

- The subjects are led to believe that their task is the evaluation of the dialog system. This is what they were told as the "story" for the experiment.
- The post-test questionnaire asked about potential misunderstandings or the subjective difficulty of the exercises, together with free-text answer fields. This further contributes to the data collection for the experimental sessions.

- Even though the subjects evaluated a non-existing system, the observations and expectations of the subjects can be potentially used for guiding the development of dialog systems in mathematics.

3.2.4 The Course of an Experimental Session

The subject was welcomed by the experimenter, who led her or him into the subject room, identified as "usability lab" by a tag on the door. The subject sat down at a desk in this room with a computer running the dialog tutoring system interface. The experimenter verbally gave an overview of the instruction material, presented on paper printouts to the student. Furthermore, he mentioned that the experiments are voluntary and that subjects may quit the experiment at any point in time. The experimenter emphasized that the system expects input in the form of natural German language, formulae, and a mixture of both. The subject was informed that the experimental sessions are recorded, and that the participants are asked to "think aloud" during the interaction with the system, if possible. The experimenter also pointed to the pre-questionnaire on the computer, which had to be filled out before starting the interaction, and the study material on paper, which was to be studied for at most 25 minutes. During the filling out of the pre-questionnaire and reading of the study material, the experimenter left the room and entered the "wizard room". The experimenter had an audio and a one-way video connection to the subject and potentially answered questions from the subject. The main task of the experimenter was now to control the timing of the experimental session.

Instructional Material The subjects had approximately 5 minutes to fill out the pre-test questionnaire and 25 minutes to read through all the instruction material before the interaction with the system. The pre-test questionnaire asks information about subject of studies, age, previous knowledge about computer systems, dialog systems and mathematics. It was presented as a TeXmacs document [19] on the subject's computer screen, and filled out and stored in electronic form.

General information about the experiment was presented on a one-sided A4 paper sheet as presented in Appendix A.1. The course of the experimental session was outlined on two one-sided A4 paper sheets as presented in Appendix A.2. This information includes an overview over the different phases of the experiment (pre-test questionnaire, reading the material, system interaction, and post-test questionnaire) and the proof exercises the participants had to solve.

The mathematics study material "Mengen und Relationen - Eine Auffrischung" is a collection of definitions and methods in the domain of set theory and relations.

Its purpose is to provide some readily available background of the domain, which the participants would be likely to have in a tutoring situation with a dialog system. Essentially, the material was taken from [12], and modified for coherence with the notation employed in the dialog system. The material was both given to the students in the form of a paper printout and as a TeXmacs document on their computer screen. The text is presented in Appendix A.3. The original handout has a slightly different page layout and fits on four pages.

It is well known that different representations of the same problem can make the problem easier or harder to solve (this is the well-known "representation problem" in problem solving). Therefore, the study material was chosen in order to reflect current mathematical practice, and concentrated on those concepts that were relevant for the task.

Even though the material was selected in the hope to influence the subjects as little as possible in their approach to the exercises, the subjects showed a strong trend to favor a certain mathematical style seemingly related to this study material. The hypothetical influence of the study material was tested by presenting some different study material to approximately half of the participants and compare both the problem solving approaches and the verbosity of the participants under the two different conditions. The second version of the mathematics study material - shown in Appendix A.4 - was created by explicitly wording large parts of the formulae that appeared in the first version of the study material. For example, the definition of the concept of a subset is presented in the first version of the study material as shown in Figure 3.3, whereas it appears in the second version as shown in Figure 3.4.

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, B are sets and $\forall x(x \in A \Rightarrow x \in B)$ holds, then A is called a subset of B. This is denoted as $A \subseteq B$."

Figure 3.3: Subset definition (1)

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, B are sets and it holds that any element of A is also an element of B, then A is called a subset of B. This is denoted as $A \subseteq B$."

Figure 3.4: Subset definition (2)

Another difference between the two versions is that in the second version, some of those definitions and explanations that were not directly relevant to the exercises were dropped, like properties of binary relations such as linearity, connectivity or trichotomy, for example. Importantly, the concept of distributivity of $\wedge$ over $\vee$ in logic was not presented in the second version of the study material.

There were no explicit instructions on how to conduct mathematical proofs. The purpose of this is to capture the expectations of the wizards (and not to test our own material).

The sample proof During the very first experiments, nearly all of the participants used extremely few words to explain their attempts to solve the proof exercises. Since the study was aimed at investigating linguistic phenomena that occur when mixing natural language with mathematical expressions, the experiment seemed to fail these goals. After seven experimental runs, the mathematics study material was extended by a "sample proof", written in an explicit style, that was aimed at encouraging the students to use natural language. This sample proof (see Appendix A.4) was presented to all participants in the study, except the first seven.

Informations about the Tutoring System Interface Basic instructions for the use of the TeXmacs interface of the dialog system were presented on two one-sided A4 paper sheets, where the second page mainly consisted of a symbol table illustrating the LaTeX commands that could be used to create mathematical symbols. This document is shown in Appendix A.5. The first subjects in the experiment showed a very heterogeneous performance in the mastery of the interface. Consequently, the decision was taken to personally instruct participants about the use of the interface before starting the proof exercises. This usually took less than 5 minutes, and provided the participants with a more homogeneous background knowledge about the interface.

3.2.5 Mathematical Domain

The exercise set for the experiment (Figure 3.5) contains a sequence of exercises about relations. These exercises focus on properties of relations and operations on relations, in particular their inverses and relation composition.

The exercises W, A, B and C build upon each other and can be solved using different mathematical styles. Therefore, they are suited to study variations in the student's and wizard's approaches towards them. Their conceptual background is very similar, which makes reading the accompanying study material "manageable" in the scope of a two hours session.

Exercise W

Assume R and S are relations on an arbitrary set M. It holds that:

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

Now conduct the proof interactively with the system; you have 15 minutes. Please proceed stepwise.

Exercise A

Assume R, S and T are relations on an arbitrary set M. It holds that:

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

Now conduct the proof interactively with the system; you have 25 minutes.

Exercise B

Assume R, S and T are relations on a set M. It holds that:

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

Now conduct the proof interactively with the system; you have 25 minutes.

Exercise C

Assume R and S are relations on a set M. It holds that:

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

Now conduct the proof interactively with the system.

Exercise E

Assume R is an asymmetric relation on a set M. Show: If R is not empty (i.e. $R \neq \emptyset$), then it holds that:

$R \neq R^{-1}$

Now conduct the proof interactively with the system.

Figure 3.5: Proof exercises in the Wizard-of-Oz study

Exercise W requires the definitions of inverse and relation composition, as well as the commutativity of the logical operator $\wedge$. This is a rather short exercise that serves as warming up exercise for the subjects to get acquainted to the definitions.

Exercise A is the distributivity of $\circ$ over $\cup$. Together with the definitions of $\circ$ and $\cup$ this problem can be reduced to the distributivity of $\wedge$ over $\vee$. But this problem can also be tackled by using the extensionality principle and case splits (and also analogy).

Exercise B is basically combined out of the preceding problems. Even though it is not necessary to solve the problem, an additional lemma $(R^{-1})^{-1} = R$ can be applied to make the proof shorter and supposedly more elegant.

Finally, *Exercise C* is a theorem which is not generally valid, hence it cannot be proven in the general case. Those students who managed to get through to this exercise where challenged to find out that the theorem is not generally valid, and argue why. The theorem is valid only if the relation S is symmetric.

During the course of the experiment, we observed students who - even though satisfying the formal criteria - struggled enormously even with the basic understanding of the concepts present in the study material and the exercises. In these cases, switching to the next, even more difficult exercise after approaching a time-limit was perceived as frustrating both by the participants and wizards, and did not help in achieving a constructive dialog. Therefore, a "special" exercise was devised (*exercise E*), which is based on the fundamental understanding of the concept of a relation and can be solved in very few steps. If substantial difficulties kept the participants from reaching the proof goal of the first exercise, they were asked to do this exercise instead of exercises A, B and C.

The basic idea behind exercise *E* is that if the asymmetric relation R is nonempty, it contains at least one pair (a,b), but not its inverted counterpart (b, a). Since R^{-1} contains the inversion of this pair (b, a) - but R does not - R and R^{-1} cannot be equal.

3.3 Related Work

Wizard-of-Oz Study on Socratic and Didactic Tutoring Another study in the Wizard-of-Oz paradigm has been carried out at the University of Pittsburgh. Its aim is the comparison of the relative effectiveness of Socratic and Didactic tutoring styles. The domain of tutoring was basic electricity and electronics (BE&E). Twenty students were randomly assigned to the Socratic or Didactic conditions, in both of which they were tutored by a post-doc at the Learning Research and Development Center at the University of Pittsburgh via a chat window. From the tutor's point of view, the Socratic method was characterized by an emphasis on eliciting information from the student but by avoiding as much as possible to give away information. On the other hand, the Didactic condition was characterized by an initial explanation of the material to be learned. The role of questions in Didactic method is defined as drawing the student's attention to information the tutor has already explained. The content of the material to be learned by the students was formalized as 47 rules about electricity and electronics. The mastery of the rules exhibited by the students was documented in a pre- and a post-test. Learning gains were determined according to these mastery scores. The scores were compared w.r.t the tutorial conditions (Didactic and Socratic), and also correlated to the extent to which the features of the Didactic and Socratic strategies were actually present in the dialogs.

The results of the comparison were of "little statistical power" (cf. [43] p. 4), due to the limited number of participants. But they show a trend in favor of the Socratic tutoring strategy.

Linguistik Markers to Improve the Assessment of Students in Mathematics: An Exploratory Study The multidisciplinary *Pépite* project aims at supporting math teachers with software support to assess their students in elementary algebra. They conducted an "exploratory" empirical study with 168 students aged 15-16 from French secondary schools (cf. [38]). The students were asked to say if some algebraic properties are true or false and to justify their answers in their own words. The Pépite project investigates the automatic classification of these justifications. A focus is on "mathural" language, i.e. "language created by students that combines mathematical language and natural language". The aim is to investigate the link between the discursive modes of the students' justifications and their level of development in algebra thinking.

Evaluating Dialog Schemata with the Wizard-of-Oz Computer-assisted Algebra Tutor The Wooz tutor developed by the North Carolina A & T algebra tutorial dialogue project is a computer program that "mediates keyboard-to-keyboard tutoring of algebra problems" (cf. [31]). These problems are high school and college algebra problems. The program allows Wizard-of-Oz studies where a tutor and a student collaboratively construct answers in the form of equations to symbolic manipulation problems and word problems. The tutor is offered pre-computed answers, and can choose whether to directly pass this answer on to the student or to modify it. Transcripts of the tutorial dialogs were collected, both from Wooz-assisted and non-Wooz-assisted tutorial sessions. The aim is to assess the completeness and effectiveness of the tutorial schemata and language integrated into the Wooz tutor, indicated by the number of times the tutor deviated from the proposed answers.

3.4 Discussion

The challenge posed by the design of the experiment was the collection of "realistic" tutorial dialogs in mathematics. There was the danger that the instructions given to the students and the wizard will direct their behavior towards the expectations by the so-called "wishful" experimenter. An example for an independent variable that potentially effects the behavior of the subjects is the study material. Its effect was assessed by using two sets for two different groups of subjects. On the other hand, it would not have been desirable to omit the mathematical study material. In order to simulate a system for teaching, it was important to have a domain where the students can not recall a proof for the given problem by heart, since this would constitute a memory experiment rather than an experiment on reasoning. Therefore, at least some mathematical concepts that were new to most of the participants had to be presented to them. The possibility to learn something new was also a motivational factor for the participants in the study.

In a way, the four wizards were also subjects in the experiment, since their understanding of mathematical tutoring was subject of investigation, too. This concerns in particular the ratings that they had to provide for the proof steps. In order not to artificially influence the wizards, they did not receive training in any particular tutoring style (such as the Didactic or the Socratic style, as discussed in Section 3.3) in the frame of the experiment.

The realization of the experiment was a demanding task. By design, the experimenter and the wizard were always two different persons. In order to standardize the instruction procedure - and to control the effect of the "wishful experimenter" - three different persons were acting as experimenters. They where equipped with the same set of instructions, which they had to present orally to the subjects, in addition to the written materials. Again another person - usually the author of this thesis - was operating the technical equipment, providing assistance to the experimenter and the wizard, and acting as an observer. Due to the technical requirements, like the two rooms with special recording equipment, and the large number of persons involved, only one experimental session could be held at the same time.

But the work of preparation and carrying-out was rewarded by the successful experiment, which - to the knowledge of the author - has a unique design and embarks onto a mathematical domain that has not been subject of a Wizard-of-Oz study in tutoring before.

Chapter 4

DiaWOz-II: A Tool for Wizard-of-Oz Studies in Mathematics and Engineering

DiaWOz-II was developed for a *Wizard-of-Oz* study in the year 2005. It is the second software environment developed for Wizard-of-Oz studies in the DIALOG project at Saarland University. Its predecessor system DiaWOz was used in a Wizard-of-Oz study in 2003. In both cases of DiaWOz and DiaWOz-II, the purpose is to provide a software infrastructure for Wizard-of-Oz experiments to investigate tutoring in mathematics and engineering. DiaWOz-II consists of a client-server architecture, where one client is operated by the student, and one client is operated by the wizard emulating crucial functions of a hypothetical "dialog system". The server mediates between both clients and records the dialog events in a log-file. Furthermore, the software provides infrastructure like a spell-checker and built-in exercise texts, as well as a mechanism that controls turn-taking between the student and the tutor.

In the 2005 Wizard-of-Oz study undertaken by the DIALOG project, the student's task was to collaboratively solve mathematical exercises about sets and relations with the "dialog system". The ability to use mathematical and technical symbols and constructs motivated the choice of TeXmacs (see [19], [25]) as a building block for our software. This makes DiaWOz-II fundamentally different from the older DiaWOz system.

The following sections provide an introduction to TeXmacs, followed by a specification of the software and its architecture.

This thesis significantly contributed to the design of DiaWOz-II, the programming of the behavior and the appearance of the TeXmacs interface - as described in Sections 4.2.1 and 4.2.2 - and the implementation of the state machine for the server.

4.1 TeXmacs

The interfaces for both the participant and the wizard in our study are based on TeXmacs, a scientific text editor (cf. [19], [25]). This editor has already been adapted as an interface to a diversity of tools, most of which are computer algebra systems. Using TeXmacs as an interface for a (simulated) tutoring system is a novelty. GNU TeXmacs is a scientific text editor which was inspired by the typesetting system TeX and the *GNU emacs* text editor (cf. [25]). The documents written in the TeXmacs editor are well organized in a tree-like structure. Unlike TeX, TeXmacs provides a "wysiwyg" (what-you-see-is-what-you-get) interface that allows to directly manipulate the typeset document. A "source" mode provides a view on the description of the document in the TeXmacs markup language.

4.1.1 TeXmacs Markup Language

TeXmacs markup is a structured language. It provides the faculty to define *macros*, which are lambda expressions. The syntax of the markup language is oriented towards TeX syntax, and many TeX-commands have their counterparts in TeXmacs. External applications and programs can be integrated into a TeXmacs document in the form of a "session" macro. Such a "session" macro receives input from the document, passes it on to the application, and displays the results in the document. For example, a SCHEME session can be inserted into a TeXmacs document, as shown in Figure 4.1.

TeXmacs markup can be presented in different syntactic variants, one of which resembles XML syntax. Another variant are SCHEME expressions representing the markup. Simple TeXmacs documents can be converted to LaTeX markup.

Macro definitions can be part of a so-called "plug-in", a collection of definitions that are available to TeXmacs. These plug-ins also define the behavior of TeXmacs. For example, an alternative menu structure can be imposed on the TeXmacs interface in the form of a plug-in.

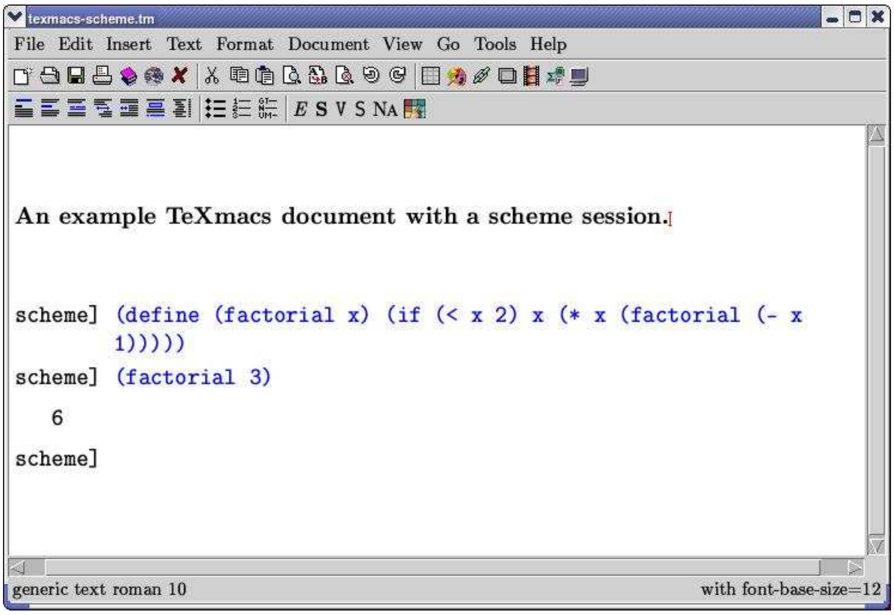

Figure 4.1: SCHEME session in TeXmacs

4.1.2 Example Macro

Macros in TeXmacs are lambda expressions that can even be applied recursively. To demonstrate this, consider the macro *factorial* (see Figure 4.2).

```
<assign|factorial|
  <macro|x|
     <if|<lesseq|<arg|x>|1>|
        1|
        <times|<arg|x>|<factorial|<minus|<arg|x>|1>>>>>>
```

Figure 4.2: TeXmacs macro defining the factorial

The macro *factorial* takes a number n as an argument and calculates the n-th factorial $n!$.

The "*assign*" macro assigns the name "*factorial*" to its macro definition. All macros definitions are tuples (of various arities), where the first component is the macro's name. The last component is the body of the macro, akin to the function body in programming languages. The other components of the macro definition are the macro's arguments. In common notation for lambda calculus, the macro definition for the factorial can be expressed as in Figure 4.3. Applying a macro to concrete arguments (and thus replacing the argument variables by their concrete instances in the body of the macro) is called "macro expansion". This corresponds to beta reduction in the lambda calculus. Evaluating the expression "<fac|3>" will thus yield the value "6".

$$factorial := \lambda x \text{ (if } x <= 1 \text{ then } 1 \text{ else } x * factorial(x-1) \text{)}$$

Figure 4.3: Lambda expression defining the factorial

4.2 DiaWOz-II as a TeXmacs Plug-in

Choosing TeXmacs as a building block for our DiaWOz-II software was motivated by the requirements imposed by the experimental design.

- TeXmacs can communicate with external systems (in this case, the other client TeXmacses via a server).
- TeXmacs is a *wysiwyg* editor.
- The TeXmacs interface can easily be customized. In this case menus were redefined, and new menu items, new commands and new keyboard shortcuts were introduced.
- Since the "dialog system" is supposed to represent a mathematical tutoring system, easy integration of mathematical symbols was crucial. Furthermore, TeXmacs allows structural constructs in the form of macros. This way, set and pair notations could be inserted into the text with only one command by the user.
- The internal representation of TeXmacs is compatible with LaTeX. This allows the use of LaTeX commands by the user. Most importantly, it also provides a suitable representation format for the empirical data in the form of a dialog corpus created in the experiment.
- TeXmacs offers advanced editing facilities like *copy and paste*, *undo* and *redo*.

4.2.1 The Student Interface

The "dialog system" is presented to the student as a window consisting of menu bars and a text field, as shown in Figure 4.4. In a second window, which is independent of the "dialog system" window, a supplementary mathematics text

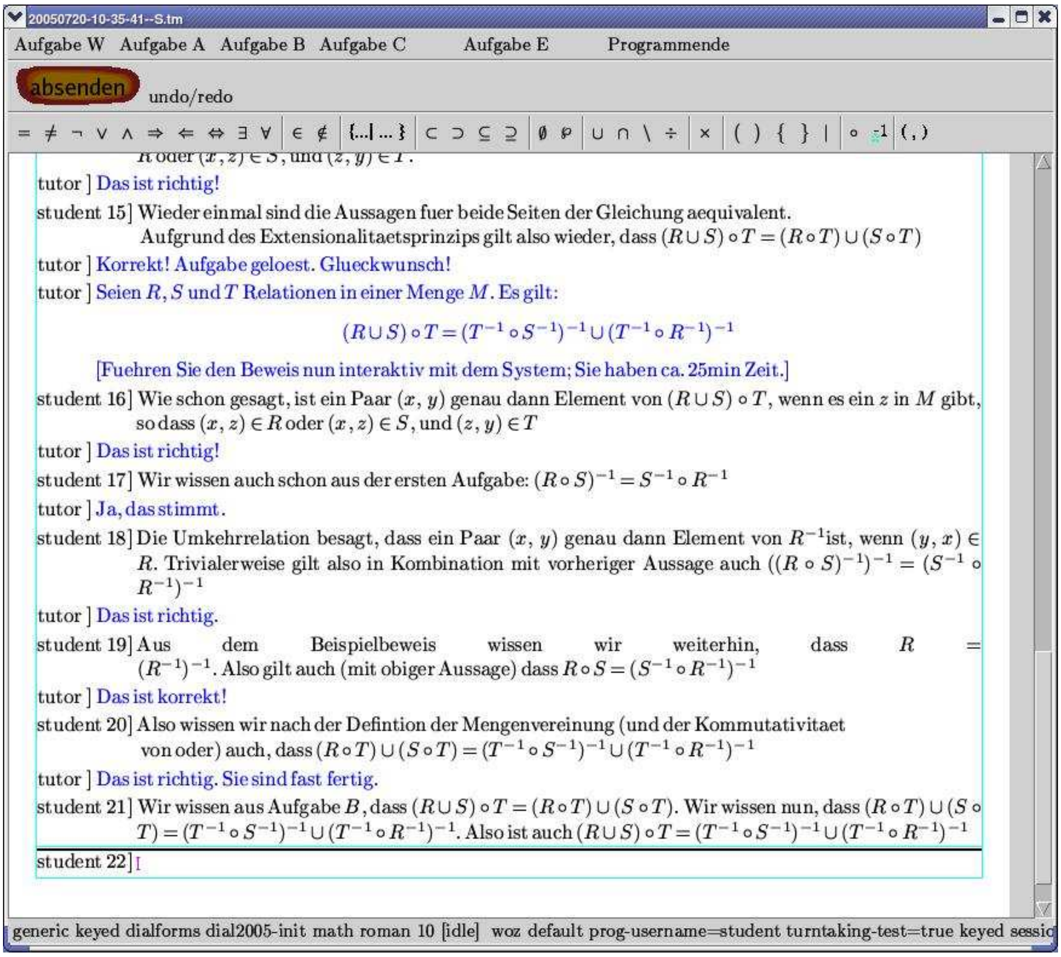

Figure 4.4: The student interface

with mathematical concepts and definitions is displayed. This text can be found in appendix A and is discussed in Section 3.2.4. In the dialog system window, the dialog history and the prompt for the current input are displayed in the same document, separated by a horizontal bar. The texts by the tutor and the student are colored differently for better readability. The student can send off messages via a dedicated button. This way, the message becomes part of the dialog history. The answers by the tutor are accompanied by an acoustic signal.

Controls

The top menu bar of the “dialog system” offers a choice of exercises. In one “development” version of the interface, the menu items representing different exercises are to be invoked in a particular predefined sequence. The wizard can decide when to move from one exercise to the next, which becomes visible to the student when the corresponding menu item flashes up. In the current interface, there is no order of succession for the exercises - this allows a restart of the system if necessary, or deviating from the given order.

A yellow and red button allows to send messages. If it is the student's turn, it has a fresh yellow color, which changes to a darker shade after the message has been sent off until the student is enabled to send a message again. Messages can also be sent via the "F12" key on the keyboard.

The third menu bar shows mathematical symbols that can be inserted into the text.

Input

The interface allows writing text with the keyboard, and in particular also LaTeX commands which are directly interpreted. Mathematical symbols can thus be created by their LaTeX commands, by additional pre-defined commands in German language, or via the menu bar. TeXmacs also allows for structured commands, like commands that create pairs of brackets for pair and for set notation. An example is the "menge" macro whose definition is given in Figure 4.5. Invoking "\menge" with the arguments "a" and "$a \subseteq A$" yields the formula "$\{\, a \,|\, a \subseteq A\}$". The arguments must not necessarily be provided when invoking the macro, they can also be filled in and modified afterward. These macros can be nested, and most importantly, they avoid missing brackets when the user writes expressions in set and pair notation.

```
<assign|menge|
    <macro|left|right|
       <left|{> <arg|left> <mid|\|> <arg|right> <right|}>
       >>
```

Figure 4.5: Macro "menge"

4.2.2 The Wizard Interface

The wizard's interface, as shown in Figure 4.6, is conceptually similar to the student's interface. Unlike the student, the wizard is not constrained in sending successive messages. On the other hand, the wizard is asked to give ratings for

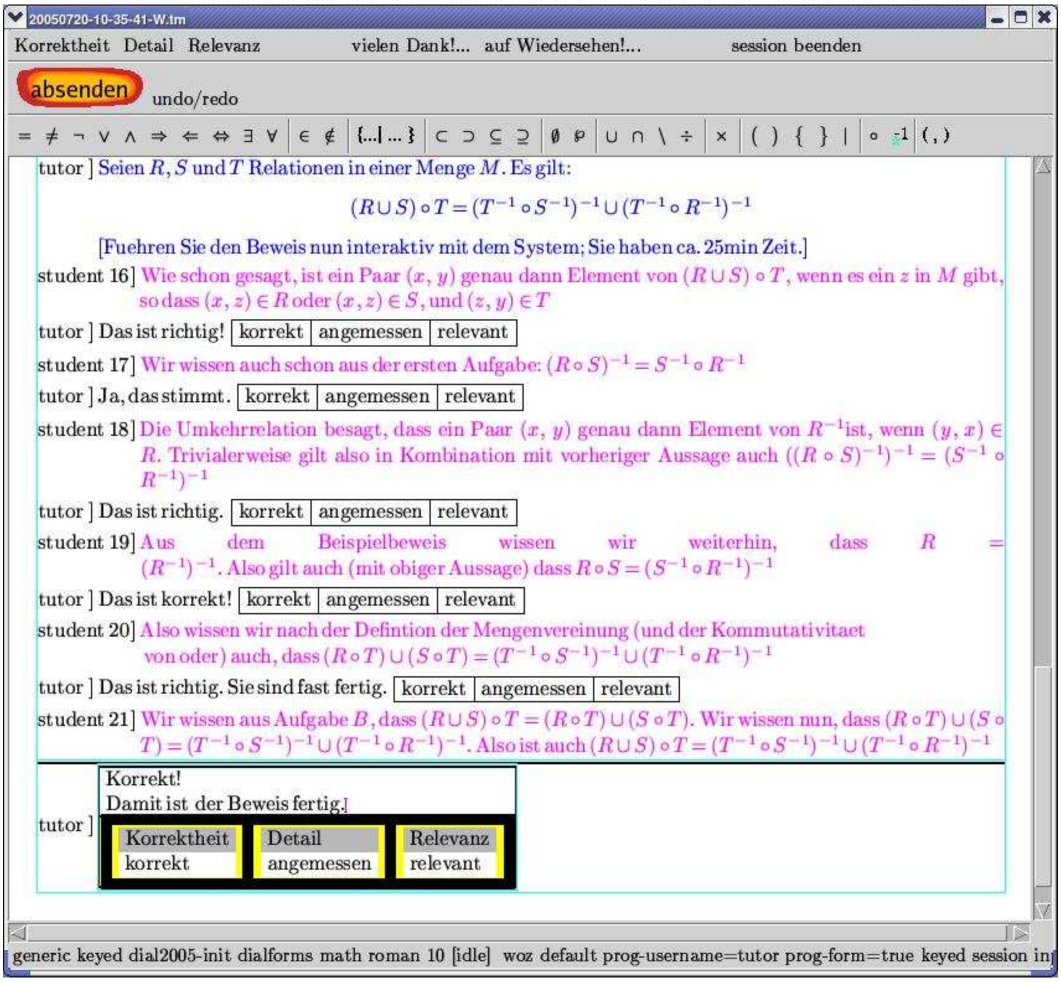

Figure 4.6: The wizard interface

each turn by the student, w.r.t. three dimensions; correctness, granularity and relevance. These slots can be filled in with the help of menus in the top menu bar. The wizard's button for sending messages is only enabled if all the slots have been filled in, in order for the wizard not to forget to perform the rating. The ratings can take on default values, in case the student's utterance does not represent a mathematical statement.

4.2.3 Architecture

DiaWOz-II consists of the two TeXmacs interfaces described before, which are connected via a server process (a classical client-server architecture). The tool allows keyboard-to-keyboard interaction between these two interfaces, as well as a log-file mechanism that records the communication events and a spell-checker. A so-called "plug-in" for TeXmacs defines the functionality of the TeXmacs client processes. This includes

- A session, i.e. a program that runs inside a TeXmacs document. In this case it is responsible for passing on communication acts to the server and displaying the incoming messages from the communication partner.
- Specifications of the TeXmacs GUI that are specific to DiaWOz-II, like menus with mathematical symbols from our domain and buttons for sending messages and starting exercises.
- Definitions of additional macros to support the user, e.g. commands in German language to create mathematical symbols.

The server is a SCHEME process that broadcasts the messages from one client to the other. These messages may contain dialog contributions from any of the connected TeXmacs clients, but also commands from the server to the clients. Thus, the server is controlling the whole architecture via messages. The server has different states corresponding to different phases of a session. These are:

- An idle phase until both student and wizard interfaces are connected.
- The log-file is created.
- Exercise solving; for each exercise first the canned exercise text appears on both student and wizard interface; then the interaction starts. The communication acts are stored in a dedicated format in the log-file. Each exercise is associated with a different state.
- Finally, the end of the session is indicated, the interfaces are shut down and the post-test questionnaire appears on the student's screen.

Messages from the clients can trigger state transitions in the server. For example, if the start of an exercise is requested by a client, a corresponding message is sent to the server. The server then changes its state, and in turn sends messages to the clients, like the exercise text to be displayed, and messages indicating which functions of the interface are now enabled.

4.2.4 Turn-Taking

The exercise solving process for each exercise is subdivided into turns. The student is allowed to make only one turn before she or he gets an answer from the tutor. Until then, the sending of new messages is disabled, i.e. the dedicated button for "sending" is displayed in a darker shade and has no effect. The tutor is not constrained in sending messages. This allows him to ask if help is needed if the student does not submit any messages for a longer period, for example.

Turn-taking is controlled by a finite state machine inside the server. On the arrival of a message at the server, it is sent to a spell-checker. If it passes, the message is broad-casted to the other client[1], and written to the log-file. If the message doesn't pass, it is sent back to the sender for correction. If the "student" has made a turn, she or he is disabled from sending new messages until the wizard makes a turn. Enabling and disabling the sending facility for the student is done via corresponding messages from the server to the student interface.

4.2.5 Spell-Checking

All messages from the student and the tutor are analyzed by a spell-checker integrated into DiaWOz-II. This is done for two purposes: Firstly, the wizard is required to write linguistically error-free texts in order not to reveal his identity. So any incorrect message that is sent from the wizard to the server is sent back to the wizard for correction. Secondly, the student's messages are refused if the number of spelling errors exceeds a threshold, with the rationale that such messages would be too difficult for a real dialog system. The spell-checking is provided by GNU Aspell, a free spelling checker software.

4.2.6 Log-File

The log-file is created as soon as a student and a wizard interface are connected to the server. It contains as a header the identity of the two conversational partners, and a time-stamp when the session was begun. Each message relevant for the tutorial dialog (thus excluding internal messages, like for the enabling of buttons, etc.) is then stored according to the message format given in Figure 4.7. Messages consisting of only a number and a time-stamp signal the start of an exercise, namely the exercise with the corresponding number. Messages that contain a sender and a document represent a dialog turn by either of the participants, or a message from the server, like the end of the session. Similar messages are also used for "internal" messages between server and clients, but they are not stored as a part of the dialog. An example for an entire log-file is presented in Appendix C.

[1]In principle it is even possible to register more than two interfaces to the server.

```
<message>  :=
  (<exc. no> <time> <sender> <document>)
| (<exc. no> <time> <sender> <document> <spellck. output>
                                         [<spellck. message>])
| (<exc. no> <time>)

<exc. no> := 0 | 1 | 2 | 3 | 4 | 5
<time>   := <dd><mm><yy><hh><mm><ss>
<sender>      := ‘‘student’’ | ‘‘wizard’’ | ‘‘tutor’’
<document> ... is any kind of document in TeXmacs markup,
               or a simple string
<spellcheck output> := (‘‘SPELLOUT’’ <wordlist>)
<wordlist>    := | <word> wordlist
<spellcheck message> ... a string containing a message from
                        the spell-checker to the <document>
                        author
```

Figure 4.7: Log-file message format

4.3 Discussion

TeXmacs as a state-of-the-art scientific editor offers many advanced features, which makes it a useful component for DiaWOz-II. The final architecture of DiaWOz-II is thus basically that of a "chat program" of connected TeXmacs interfaces enhanced by specific mechanisms. Building a dedicated "chat" program this way might be a nice alternative to existing chat tools like "icq messenger", "gaim" or "kopete" because these do not support a viable interface for formulae and technical expressions, which might be a feature appreciated by the scientific (chat) community.

Besides being a chat program, the architecture offers additional features in the form of a spell-checker and a log-file mechanism. On top of that, it provides a tutoring-system specific mechanism enforcing a (rather simplistic) dialog structure, in the form of alternating dialog turns that are grouped according to the exercises.

The most difficult part of the design in this respect was to "tame" the complex and flexible behavior of TeXmacs. Standard TeXmacs allows the user to edit parts of the document in a multitude of ways, which was not desired for the dialog history, for example. On the other hand, it was desirable to allow the copy & paste facility to copy from the history (but not to delete from it). Therefore, some standard editing facilities had to be disabled on some parts of the document, which required additional computation. The necessity to even "restrict" TeXmacs for our experiment even more demonstrates the potential of TeXmacs as an interface component for complex systems.

Chapter 5

Analysis of the Corpus

The Wizard-of-Oz experiment resulted in a corpus that consists of 37 log-files, which contain the interactions of the 37 participants in our study with the "mathematics tutoring system". These dialogs recorded by the DiaWOz-II software described in Chapter 4 are in written form. The subjects were told that the "mathematics tutoring system" is capable of natural language dialog and encouraged to use natural language as well as mathematical formulae. The dialogs are accompanied by audio and video recordings of the subjects. The following sections provide an overall picture of the collected corpus, and the identification of phenomena that are related to granularity in particular.

5.1 General Observations

During the course of one experimental session, the 37 "student" participants produced on average 24.38 dialog turns - an initial standardized "interface familiarization exercise" excluded. So in total the corpus amounts to 902 dialog turns by the "student" subjects and just as many responses by the tutor. One potential subject's run had to be canceled due to a disk failure. She is not counted among the 37 participants.

5.1.1 Mathematical Style

There were two groups of subjects who received the two different versions of mathematical study material discussed in Section 3.2.4, with 20 subjects in the first group and 17 in the second. There is a potential influence of the mathematics study material on the mathematical "style" in which the subjects tried to solve the exercises. In order to separate the influence of the study material from the influence of the tutor on the mathematical style of the subjects, the subjects in the two groups were distributed evenly on the four tutors.

Due to the design of the experiment, "exercise W" (as presented in the instructions for the participants in Appendix A.2, and also in Figure 5.3) is the only exercise which was attempted by all of the subjects. It is also the first exercise that was presented to them. There are at least two approaches to that exercise.

The first style will be referred to as "pure term-rewriting" style. Starting from one side of the equation to be proven, the term - or a sub-term of the term - is replaced by a logically equivalent term, based on logic equivalence or on mathematical definitions. This is done successively in order to obtain the term on the other side of the equation. Sometimes, both sides of an equation are simultaneously worked on until they represent the same term. An example for such a proof of exercise W is given in Figure 5.3.

"*The extensionality principle for sets states that two sets A,B are equal iff they contain the same elements,*

$$A = B \Leftrightarrow \forall x(x \in A \Leftrightarrow x \in B)."$$

Figure 5.1: Extensionality principle

"*An important relationship between inclusion and equality of sets follows from the extensionality principle. For sets A, B holds*

$$A = B \Leftrightarrow A \subseteq B \wedge B \subseteq A$$

From this formula one extracts an often used proof principle: To show equality of two sets A, B, one has to prove the two inclusions $A \subseteq B$ and $B \subseteq A$."

Figure 5.2: Proof principle based on extensionality

The second style makes use of the definition of equality by the extensionality principle. This principle was presented to the subjects in the study material, and is stated in Figure 5.1. In the case of equality between sets, equality is shown by demonstrating that an element of the first set is also an element of the second set, and vice versa. This establishes a subset relation between the first set and the second. The study material points to an often used proof principle that can be derived from the extensionality principle, which is presented in Figure 5.2. The dialogs in the corpus contain examples where equality between sets is shown by two subset relations, but also examples where the extensionality principle is used directly. An example proof for exercise W with the extensionality principle is found in Figure 5.4.

tutor] Seien R und S Relationen in einer beliebigen Menge M. Es gilt:

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

[Fuehren Sie den Beweis nun interaktiv mit dem System; Sie haben ca. 15min Zeit. Bitte gehen sie moeglichst Schrittweise [sic] vor.]
"Let R and S be relations in an arbitrary set M. It holds: $(R \circ S)^{-1} = S^{-1} \circ R^{-1}$
Now conduct the proof interactively with the system; you have approximately 15 minutes. Please proceed stepwise where possible."

student 0] $(R \circ S)^{-1} = \{(y, x)|(x, y) \in (R \circ S)\}$
tutor] Die Aussage ist richtig.
"This statement is correct."

student 1] $(R \circ S)^{-1} = \{(y, x)|\exists z(z \in M \wedge (x, z) \in R \wedge (z, y) \in S\}$
tutor] Auch diese Formel stimmt.
"This formula is also correct."

student 2] $(R \circ S)^{-1} = \{(y, x)|\exists z(z \in M \wedge (z, x) \in R^{-1} \wedge (y, z) \in S^{-1}\}$
tutor] Das ist richtig. Sie sind auf einem guten Weg.
"This is correct. You are on a good way."

student 3] $S^{-1} \circ R^{-1} = \{(x, y)|\exists z(z \in M \wedge (x, z) \in S^{-1} \wedge (z, y) \in R^{-1})\}$
tutor] Das ist ebenfalls korrekt.
"This is also correct."

student 4] $S^{-1} \circ R^{-1} = \{(y, x)|\exists z(z \in M \wedge (y, z) \in S^{-1} \wedge (z, x) \in R^{-1})\}$
tutor] Sehr gut. Damit ist Ihr Beweis komplett.
"Very good. With that your proof is complete."

student 5] $(R \circ S)^{-1} = \{(y, x)|\exists z(z \in M \wedge (y, z) \in S^{-1} \wedge (z, x) \in R^{-1})\} = S^{-1} \circ R^{-1}$
tutor] Ihre Loesung ist vollstaendig und richtig. (...)
"Your solution is complete and correct."

Figure 5.3: Proof of exercise W in "rewriting" style

student 1]	Sei $(a,b) \in M \times M$. Nach dem Extensionalitaetsprinzip ist zu zeigen, dass $(a,b) \in (R \circ S)^{-1}$ genau dann gilt, wenn $(a,b) \in S^{-1} \circ R^{-1}$. *"Let $(a,b) \in M \times M$. According to the extensionality principle it is to be shown that $(a,b) \in (R \circ S)^{-1}$ holds if and only if $(a,b) \in S^{-1} \circ R^{-1}$."*
tutor]	korrekt. *"correct."*
student 2]	Sei also $(a,b) \in (R \circ S)^{-1}$. Laut Definition der Umkehrrelation gilt dann auch $(b,a) \in (R \circ S)$. *"Thus let $(a,b) \in (R \circ S)^{-1}$. According to the definition of the inverse relation then holds $(b,a) \in (R \circ S)$."*
tutor]	korrekt. Weiter so! *"correct. Continue like that!"*
student 3]	Nach Definition des Relationenproduktes gibt es ein $z \in M$ mit $(b,z) \in R \wedge (z,a) \in S$. *"According to the definition of the relation product there exists a $z \in M$ with $(b,z) \in R \wedge (z,a) \in S$."*
tutor]	korrekt. *"correct."*
student 4]	Also gilt nach Definition der Umkehrrelation $(z,b) \in R^{-1} \wedge (a,z) \in S^{-1}$. Oder $(a,z) \in S^{-1} \wedge (z,b) \in R^{-1}$ wegen der Kommutativitaet von $\wedge$. *"Therefore it holds, according to the definition of the inverse relation, $(z,b) \in R^{-1} \wedge (a,z) \in S^{-1}$. Or $(a,z) \in S^{-1} \wedge (z,b) \in R^{-1}$ because of the commutativity of $\wedge$."*
tutor]	Richtig. Sie sind kurz vor dem Ziel! *"Correct. You are close to the goal!"*

Figure 5.4: Proof of exercise W based on the extensionality principle

student 5]	Die Definition des Relationenproduktes liefert nun, dass auch $(a,b) \in S^{-1} \circ R^{-1}$ gilt, was zu zeigen war. *"The definition of the relation product now yields that also $(a,b) \in S^{-1} \circ R^{-1}$, what was to be shown."*
tutor]	Das war schon sehr gut. Leider haben Sie bisher nur eine Implikation bewiesen. *"That was already pretty good. Unfortunately you have only shown one implication until now."*
student 6]	Kann ich in einem Schritt die Aequivalenz beweisen oder muss ich zwei Implikationen beweisen? *"Can I prove the equivalence in one step, or do I have to prove two implications?"*
tutor]	Der Beweis ist in einem einzigen Schritt moeglich. *"The proof is possible in one single step."*
student 7]	Also nach 6 Schritten, die ich bisher getan habe noch einen? *"So after the 6 steps that I have done so far, another one?"*
tutor]	Sie haben genau einen Schritt gemacht, der laut ihrer Beschreibung lediglich eine Implikation und keine Aequivalenz bedeutet. *"You have done exactly one step, that according to your description signifies an implication and not an equivalence."*
student 8]	Sei $(a,b) \in M \times M$. Nach dem Extensionalitaetsprinzip ist zu zeigen, dass $(a,b) \in (R \circ S)^{-1}$ genau dann gilt, wenn $(a,b) \in S^{-1} \circ R^{-1}$. *"Let $(a,b) \in M \times M$. According to the extensionality principle it is to be shown that $(a,b) \in (R \circ S)^{-1}$ holds if and only if $(a,b) \in S^{-1} \circ R^{-1}$."*
tutor]	korrekt. *"correct."*

Figure 5.4: Proof of exercise W based on the extensionality principle, continued

student 9]	Sei also $(a, b) \in (R \circ S)^{-1}$. Laut Definition der Umkehrrelation gilt dies genau dann, wenn auch $(b, a) \in (R \circ S)$. *"Let thus $(a, b) \in (R \circ S)^{-1}$. According to the definition of the inverse relation, this holds if and only if $(b, a) \in (R \circ S)$."*
tutor]	Sehr schoen. Genau dieser Schritt war zu aendern. Glueckwunsch zum Loesen dieser Aufgabe! *"Very nice. Exactly this step was to be changed. Congratulations for solving this exercise!"*
student 10]	Danke! *"Thank you!"*

Figure 5.4: Proof of exercise W based on the extensionality principle, continued

A first observation from the experiment is that two of the tutors suggested a proof based on the extensionality principle as a master solution. Only one tutor proposed a rewriting style proof. The fourth tutor mixed these two approaches. The master solutions are found in appendix B.

	Group 1	Group 2
Pure rewriting approach by student	12	0
Extensionality principle approach by student	1	12
Extensionality principle approach suggested by tutor	4	3
Other approaches/ambiguous	3	2

Table 5.1: Proof styles for exercise W. Groups 1 and 2 received the first and second version of the study material, respectively.

Interestingly, the experiment indicates an influence of the mathematical study material on the mathematical style employed by the subjects. The figures showing how many members of the two groups were using a particular style - also including those cases where it was not possible to unambiguously assign a style - are presented in Table 5.1. In the group of 20 who were given the first set of study material, the majority (12) attempted exercise W in a pure rewriting style,

	Group 1	Group 2	All subjects
Average number of NL words per session	75.05	119.47	95.46
Standard deviation	72.01	81.90	79.84
Maximal number of NL words per session	220	347	347
Minimal number of NL words per session	0	10	0
Average number of NL words per turn	3.27	4.58	3.92

Table 5.2: Word count (of words in natural language) for the dialogs. Groups 1 and 2 received the first and second version of the study material, respectively.

whereas the majority of students in the second group (12) attempted a proof based on the extensionality principle. Some of the students who did not have a clear idea on how to proceed were guided by the tutor in choosing a particular style. Most interestingly, when the tutors suggested an approach to solve the first exercise, none of them instructed the students to use a pure “term rewriting” approach, even though this style was quite common among the subjects.

5.1.2 Verbosity

Table 5.2 shows how many words in natural language - as opposed to formulae - the subjects in the study produced per session and per dialog turn, depending on whether they were given the first or the second set of study material. The most extreme cases were two students in group 1 who did not produce any word during the entire session. They approached the exercises in a pure rewriting style, i.e. they proved an equality by reformulating the term on one side of the equality until the equivalence with the term on the other side was obvious. This style does not make any words necessary, but it often led to very long, inflated formulae. One example of such a formula from the corpus is displayed in Figure 5.5.

student] $(R \circ S)^{-1} = \{(x,y)|(y,x) \in \{(x,y)|\exists z(z \in M \wedge (x,z) \in R \wedge (z,y) \in S)\}\}$

tutor] okay, geht aber einfacher.
“okay, but could be done simpler.”

Figure 5.5: Unnecessarily long formula

Here again, it should be noted that the subjects were equally instructed on the “natural language character” of the “system”. Nevertheless, the standard deviation in the number of words in natural language per session is very high, documenting the variety in the behavior of the subjects.

An analysis of the ratio between words in natural language and formulae will further illustrate the issue of the student's verbosity, but this further work is outside the scope of this thesis.

5.1.3 Effect of the Interface on Mathematical Style

The DiaWOz-II interface provides support to easily construct formulae; by providing structured constructs, by the copy & paste facility, and by offering different modes for entering mathematical symbols. So in contrast to written natural language input, inputting and manipulating formulae was encouraged by the support offered by DiaWOz-II . This effect might have been increased by the instructions on the interface the students received, since mastery of the creation of mathematical symbols was one of the main issues. The initial idea was that the easier the students could handle the software, the more they would be able to concentrate on the dialog. On the other hand, this might have led to a relative preference of formula language compared to natural language.

Some students exploited the possibilities of the interface to the very extreme. This is reflected in the huge formulae they constructed with high accuracy. Even the more, formulae were nested into each other, like in the example given in Figure 5.6. Most probably, DiaWOz-II's facilities encouraged the students to use a pure rewriting style, where proof steps were performed by replacing (sub)formulae in the current assumption or goal by an equivalent formula, usually according to definitions or logical equivalence.

Another factor of the software that potentially influenced the dialogs is speed. The long waiting time for answers was the most prominent criticism from the participants in the post-test evaluation. It is possible that the students developed the hypothesis that writing larger texts leads to longer waiting times or makes

student]	$\{(x,y)\|(x,y) \in \{(x,y)\|\exists z(z \in M \wedge (x,z) \in R \wedge (z,y) \in T)\} \vee (x,y) \in \{(x,y)\|\exists z(z \in M \wedge (x,z) \in S \wedge (z,y) \in T)\}\} = \{(x,y)\|\exists z(z \in M \wedge (z,y) \in T \wedge ((x,z) \in R \vee (x,z) \in S))\}$
tutor]	Ja. Das ist eine gute Vereinfachung. *"Yes. This is a good simplification."*

Figure 5.6: Example of a long formula

the analysis harder for the system. Furthermore, each keystroke in the DiaWOz-II interface had a small but noticeable delay, which made writing clumsy. This might also account for a disadvantage for writing natural language text rather than formulae, which are generally more concise w.r.t. the number of symbols they require.

As a consequence of the particular mathematical style based on rewriting, assisted by copy & paste facilities, the messages were very hard for the tutors to assess. Some formulae spanned several lines with several levels of nested brackets.

5.1.4 The Subject's Verdict

The usability evaluation of the simulated dialog system served as background story for the participants. However, the opinions collected this way highlight those aspects that are of importance for these potential dialog system users.

On average, the system scored a grade of 2.5 for the question: "Is the system helpful for solving the exercises?" on a scale between one (very helpful) and five (not helpful), with a standard deviation of 1.0.

Most criticism by the participants concerned the speed of the dialog system. More concretely, 16 participants complained about slowness of the system or long waiting times in the post-test evaluations forms. Further two participants explicitly mentioned the slowness of the interface, which had a noticeable delay for each keystroke. On average, the system scored 3.6 for the question: "Is the interaction with the system fluent or clumsy?", where gradings range from one (very fluent) to five (very clumsy).

5.2 Observations relevant to Granularity

The interactions between the students and the simulated system contain several passages that illustrate different aspects of granularity as identified in Chapter 2. They show that all three dimensions abstraction, explicitness and cognitive effort were indeed present in the interactions, and very often led the tutor to choose dedicated tutorial strategies, like prompting for more information. The tutors were asked to evaluate the granularity of the mathematical statements by the subjects during the experiment. In the 37 dialogs, the tutors rated 20 dialog turns as "too detailed" and 51 dialog turns as "too coarse-grained". This makes an average of 1.92 "not appropriate" dialog turns per dialog w.r.t. granularity. This shows that granularity was indeed an issue.

5.2.1 Abstraction, Explicitness, Cognitive Effort

Figure 5.7 shows a dialog fragment where two different levels of abstraction appear.

student 33]	sei $(x, y) \in (R \cup S) \circ T$, dann $(x, z) \in (R \cup S) \wedge (z, y) \in T$ *"let* $(x, y) \in (R \cup S) \circ T$, *then* $(x, z) \in (R \cup S) \wedge (z, y) \in T$"
tutor]	Ja, ein solches z gibt es dann. *"Yes, then such a z exists."* korrekt | angemessen | relevant
student 34]	dann $(x, z) \in R \vee (x, z) \in S$ *"then* $(x, z) \in R \vee (x, z) \in S$"
tutor]	Diese Aussage stimmt. *"This statement is correct."* korrekt | angemessen | relevant
student 35]	was folgt aus $(A \vee B) \wedge C$? *"what follows from* $(A \vee B) \wedge C$?"
tutor]	Dann gilt zum Beispiel $(A \wedge C) \vee (B \wedge C)$ *"For example, it holds that* $(A \wedge C) \vee (B \wedge C)$"
	korrekt | angemessen | relevant
student 36]	dann gilt also $((x, z) \in R \wedge (z, y) \in T) \vee ((x, z) \in S \wedge (z, y) \in T)$ *"then it holds that* $((x, z) \in R \wedge (z, y) \in T) \vee ((x, z) \in S \wedge (z, y) \in T)$"
tutor]	Ja, das gilt. *"Yes, this holds."* korrekt | angemessen | relevant

Figure 5.7: Example illustrating abstraction. Ratings by the tutor read as "korrekt": correct, "angemessen": appropriate, "relevant": relevant.

The student is manipulating expressions that deal with pairs as members of relations. Then in Utterance "student 35", the participant switches to a more abstract representation by replacing the set-theoretical expressions by variables A, B and C. The student then instantiates the general schema of "distributivity" with more "concrete" expressions again in Utterance "student 36". Figure 5.8 shows a dialog fragment that illustrates the role of explicitness. The subject has performed a transformation according to the "distributivity law", but does not name it. The wizard prompts the subject to name the applied distributivity property explicitly. The wizard has indicated that he considers Utterance "student 19" as

student 13]	$(R \cup S) \circ T$ ist dann $\{(x,y) \mid \exists z(z \in M \wedge ((x,z) \in R \vee (x,z) \in S) \wedge (z,y) \in T)\}$. "$(R \cup S) \circ T$ *thus is* $\{(x,y) \mid \exists z(z \in M \wedge ((x,z) \in R \vee (x,z) \in S) \wedge (z,y) \in T)\}$." (...)
student 19]	$(R \circ T) \cup (S \circ T)$ ist dann $\{(x,y) \mid (\exists z(z \in M \wedge (x,z) \in R \wedge (z,y) \in T)) \vee ((\exists z(z \in M \wedge (x,z) \in S \wedge (z,y) \in T))\}$ "$(R \circ T) \cup (S \circ T)$ *thus is* $\{(x,y) \mid (\exists z(z \in M \wedge (x,z) \in R \wedge (z,y) \in T)) \vee ((\exists z(z \in M \wedge (x,z) \in S \wedge (z,y) \in T))\}$"
tutor]	korrekt. Koennen Sie auch angeben, nach welchem Gesetz Sie Eingabe 13 zur aktuellen Eingabe 19 umgeformt haben? *"correct. Can you also indicate, according to which law you have transformed input 13 to the current input 19?"* korrekt | zu grobschrittig | relevant
student 20]	"Distributivgesetz" *"distributivity law"*

Figure 5.8: Example illustrating explicitness. Ratings by the tutor read as "korrekt": correct, "zu grobschrittig": too coarse-grained, "relevant": relevant.

"too coarse-grained", which refers to granularity.

Finally, Figure 5.9 shows a discourse fragment with a statement by the student where the second equality is not immediately obvious. The wizard has marked this step as "too coarse-grained". In his response to the student, he qualifies this step as "too quick".

5.3 Conclusion

This exploratory study has led to a corpus rich in different phenomena. The general analysis shows that the corpus contains different mathematical approaches to the same set of exercises. In many cases, the granularity of the proof steps was subject of criticism by the tutor, and all three dimensions of granularity discussed in Chapter 2 are present in the corpus.

The corpus also indicates that the instructions had an influence on the mathematical style employed by the subjects, as well as the interface in form of DiaWOz-II. It allowed the students to construct formulae that were very difficult to analyze for the human tutors, which means that the tutors often overlooked mistakes. An

student 2]	Weiter: $(R \circ S)^{-1} = \{(x, y) \mid (y, x) \in R \circ S\}$ $= \{(x, y) \mid \exists z(z \in M \wedge (x, z) \in R^{-1} \wedge (z, y) \in S)^{-1}\}$ $= R^{-1} \circ S^{-1}$ Geht das so? *“Furthermore:* $(R \circ S)^{-1} = \{(x, y) \mid (y, x) \in R \circ S\}$ $= \{(x, y) \mid \exists z(z \in M \wedge (x, z) \in R^{-1} \wedge (z, y) \in S^{-1}\}$ $= R^{-1} \circ S^{-1}$ *Does it work that way?”*
tutor]	Das geht ein wenig schnell. Woher nehmen sie die zweite Gleichheit? *“This is going a bit fast. Where do you take the second equality from?”* korrekt | zu grobschrittig | relevant

Figure 5.9: Example illustrating cognitive effort. Ratings by the tutor read as “korrekt”: correct, “zu grobschrittig”: too coarse-grained, “relevant”: relevant.

automated mechanism as part of an intelligent tutoring system might be more accurate in evaluating these syntactically blown-up formulae. But on the other hand, this style might just be an artifact of the particular experimental setup.

Chapter 6

Granularity Evaluation Framework

The granularity evaluation framework is a software component developed specifically for this thesis. It is a generic mechanism that allows the annotation of proof steps in a proof with parameters. It provides the infrastructure of how the parameters are assigned and combined in the form of *Evaluation Schemata*. The mechanism itself does not, however, embody a specific hypothesis for the evaluation of granularity. It supports different *Evaluation Schemata*, which represent different hypotheses for obtaining a measure for granularity. Some examples of these will be presented in this chapter.

6.1 Design Goals

The main design goals for the construction of the framework are:

- A high degree of independence of a particular calculus or a particular mathematical assistance system.

- To provide flexible means for the evaluation of different hypotheses about granularity. These hypotheses may require counting of proof steps, modeling of learning and repetition effects, assigning parameters to proof steps according to their particular qualities, and the combination of these parameters to the evaluation of a (partial) proof. The aim is to account for a large range of different phenomena.

6.2 Proof Representation

Here we define the structure of proofs that are to be analyzed. This representation allows for deductive multi-conclusion proof arguments with contexts. Its primary purpose is to provide a data-structure for the analysis of granularity. However, it is not concerned with the correctness of proofs that can be formulated in this representation. The goal is that - as a minimal requirement - the structure of the proofs in the PSYCOP theory [40] and Gentzen's natural deduction calculus [20] can be represented, but potentially also other calculi that fit into this very generic data-structure. The proof representation is largely inspired by the *PDS* proof data structure in ΩMEGA [44].

A proof is seen as a set of proof lines, taking the role of premises, conclusions and intermediate results, and a set of justifications providing links between these proof lines. For each proof we assume that there exists a fixed set of deduction rules R. For example, this set R can instantiated by the rule set of the PSYCOP theory which is introduced in Chapter 7.

Very often in logic, deduction rules are viewed as functions defined on proof lines that yield another proof line as their function value, the conclusion of the deduction rule application. But this view excludes calculi that produce multiple conclusions in one step, like the PSYCOP deduction system.

$$\frac{A \wedge B}{A} \qquad \frac{A \wedge B}{B}$$

Figure 6.1: AND Elimination in Gentzen's natural deduction calculus

Without delving into the PSYCOP theory at this point, it needs to be mentioned that the PSYCOP deduction system contains inference rules that introduce two new conclusions each time such a rule is applied. An example is the *Forward AND Elimination* rule, which is very similar to the *AND Elimination* rule in Gentzen's natural deduction calculus [20]. The basic difference is that in Gentzen's original calculus, *AND Elimination* consists of two rules depicted in Figure 6.1, whereas the corresponding rule in the PSYCOP calculus introduces both conclusions A and B in one step. The *Forward AND Elimination* rule is presented together with the other PSYCOP inference rules in Appendix D.

For the analysis of granularity, it is desirable to view each rule application as one cognitive act performed by a human mathematician or a proof system, independent of the number of conclusions. The requirement to assign to each rule application one parameter that represents a distinctive quality of that rule application (for example a granularity measure) requires to count each rule application exactly once. Therefore, *AND Elimination* is represented as one rule application with two new conclusions in the PSYCOP calculus and two rule applications with one new conclusion each in Gentzen's calculus. This motivates the explicit representation of each deduction rule application in a proof, in addition to the proof lines justified by these.

Definition 1 (Proof)**.** A proof P is a five-tuple $(\mathcal{S}_\mathcal{P},\Lambda_P,\Psi_P,\Gamma_P,\Delta_P)$ where the components are defined as follows:

- **Mathematical statements** $\mathcal{S}_\mathcal{P}$: A set of propositions, i.e. expressions that can be either evaluated to *true* of *false*.
- **Proof lines** Λ_P: A set of proof lines together with a total mapping $\Lambda_P \rightarrow \mathcal{S}_\mathcal{P}$ that assigns each proof line to exactly one mathematical statement. A second total mapping $\mathcal{S}_\mathcal{P} \rightarrow \mathcal{P}(\mathcal{S}_\mathcal{P})$ provides the *context* for a given proof line.
- **Deduction Rule Applications** Ψ_P: A deduction rule application is a three-tuple, consisting of an identifier of a deduction rule in R, a set of proof lines (the premises), and a second set of proof lines (the conclusions).
- **Assertions** Γ_P: A collection of proof lines $\Gamma_P \subseteq \Lambda_P$, such that for all assertions $a \in \Gamma_P$, a is not a conclusion of a deduction rule application in Ψ_P.
- **Conclusions** Δ_P: A collection of proof lines $\Delta_P \subseteq \Lambda_P$, such that no conclusion $c \in \Delta_P$ is a premise of a deduction rule application in Ψ_P.

We require that each proof line in a proof is also a conclusion of a deduction rule application, unless it belongs to the assertions. That is, for all proof lines $l \in \Lambda_P$ it either holds that $l \in \Gamma_P$ or

$$\exists! j \in \Psi_P \text{ such that } j = (d_j, premises_j, conclusions_j) \text{ and } l \in conclusions_j.$$

This limits the representation to proofs where each proof line is justified by one or zero deduction rule applications. Proofs that are not closed can be represented by using a special type of justification that identifies "open" proof goals (i.e., gaps in the proof).
This allows to define a mapping $justification : \Lambda_P \rightarrow \Psi_P$ such that if $l \notin \Gamma_P$ then

$$justification(l) = j,$$

where $j \in \Psi_P$ such that $j = (d_j, premises_j, conclusions_j)$ and $l \in conclusions_j$.
Furthermore, each proof line l can be assigned a set of premises such that:

- $premises(l) = \emptyset$ if l is an assumption,
- and $premises(l) = prem$ such that $justification(l) = (d, prem, conclusions)$, otherwise.

Furthermore, a proof is stipulated to be free of cycles, i.e. there is no sequence of proof lines $l_1, l_2, ..., l_n$ in a proof such that $l_i \in premises(l_{i+1})$ and $l_1 = l_n$.

The proof structure defines a directed acyclic graph, with edges

$$E_P = \Lambda_P \cup \Psi_P$$

and vertices

$$\begin{aligned} V = \quad & \{(j, l) \in \Psi_P \times \Lambda_P |\ justification(l) = j\} \cup \\ & \{(l, j) \in \Lambda_P \times \Psi_P | j = (d_j, premises_j, conclusions_j) \text{ with } l \in premises_j\}. \end{aligned}$$

6.3 Evaluation Schemata for Proof Arguments

The aim of the framework is to annotate the deduction rule applications in a proof with parameters that characterize a particular quality of these proof steps (like this is done for the aspect of granularity in particular, as demonstrated in the following sections). The mechanism does not only allow to use constant parameters, but to define functions that determine the parameters dynamically. Collections of the detailed specifications on how given proofs are annotated with parameters are provided as *Evaluation Schemata*.

First of all, each deduction rule application is required to represent an instance of an abstract deduction rule. In Gentzen's natural deduction calculus [20], backward and forward applications of the same deduction rule are equally considered instances of that particular rule. In the PSYCOP theory, deduction rules that correspond to backward and forward directions of rules in Gentzen's natural deduction calculus are often treated differently. They are represented by different procedures, require different preconditions to be applied, and are assigned different parameters for the "availability" of these rules for human reasoners, as will be discussed in Section 7.3.1.

Therefore, the framework allows the user to define which properties uniquely define a deduction rule, like for example whether two application directions of a deduction rule should be treated the same or not. This is not a full representation of the deduction rules, but merely a collection of features that may be relevant for determining the evaluation parameters.

Definition 2 (Evaluation Schema)**.** An *Evaluation Schema* is a five-tuple ($\mathcal{R}$, μ, $\mathcal{V}$, Φ, γ), with the components defined below.

- **Abstract Deduction Rules**: $\mathcal{R}$ is a set of identifiers that represent the different deduction rules considered in the evaluation. A total mapping $\mu : \Psi \to \mathcal{R}$ provides for each deduction rule application the abstract deduction rule it represents as an instance.

  Note here that the abstract deduction rules lack a definition of their semantics. But the objective of evaluation schemata is not to deal with the correctness of a proof, but the evaluation of those quantitative or qualitative aspects of proofs that are beyond correctness. Parameters are assigned to the applications of deduction rules in a proof, which may depend on the identity of the deduction rule applied, and the arguments of that rule application in the form of premises and parameters. In their simplest form, the parameters can just be constants.

- **Values** $\mathcal{V}$: is an arbitrary set of values.

- **Deduction Rule Application Evaluation Functions** Φ: A deduction rule application evaluation function $f \in \Phi$ is a function $\Psi \to \mathcal{V}$. It provides a parameter value in $\mathcal{V}$ when applied to an instantiation of a deduction rule.

- **Evaluation Function Assignment**: γ is a total function $\mathcal{R} \to \Phi$. It assigns to each abstract deduction rule an evaluation function, which is defined on those deduction rule applications that are instances of that abstract rule.

6.3.1 The Working of the Evaluation Mechanism

An evaluation is performed on a particular proof P and an evaluation schema S. This happens in a two-step procedure:

1. Evaluation of the deduction steps in the proof: each $d \in \Psi_P$ is assigned a value $v \in \mathcal{V}$, according to the evaluation function $\gamma(\mu(d))$.

2. Combination of the single parameters for each proof step into a combined value.

Step 1 requires that for each deduction step $d \in \Psi_P$ the evaluation function $\gamma(\mu(d))$ is retrieved and applied to d. In Step 2, all evaluation values are collected. These can then be combined flexibly to obtain sums, products, maxima, minima, averages etc. The mechanism allows to either collect only parameters from those deduction steps that contribute to the conclusions (i.e. there is a path in the graph structure that connects such a deduction step to a conclusion) or from the complete proof structure. Different examples for instantiating the mechanism with concrete schemata to obtain a specific measure are presented in Section 6.4.

6.3.2 Stateful Parameters

The mechanism has been extended to supply each evaluation function in Φ with a stateful parameter. This parameter has an initial value and changes its value each time the particular evaluation function for a deduction rule is applied. A systematic (for example, exponential) decrease or increase of the parameter values can be achieved using these stateful parameters. This can be used for modeling repetition effects or learning.

Formally, a stateful parameter is a tuple (ν,δ), where $\nu \in \mathcal{V}$ is considered as an initial value, and δ is a function of type $\mathcal{V} \rightarrow \mathcal{V}$. Each stateful parameter is associated with a particular deduction rule. The value v of a stateful parameter is calculated as $v{=}\delta^t(\nu)$, where t denotes the t-th invocation of the evaluation function. The parameters t_i can be arbitrarily reset for all stateful parameters in an evaluation schema at the same time.

The concept of the evaluation function for deduction rule applications is extended for the use of the stateful parameters, such that an evaluation function is defined on both a deduction rule application $d \in \Psi$ and the value of the corresponding stateful parameter. This allows to calculate parameters that depend both on the concrete deduction rule application and on the number of times an instance of this deduction rule appears in the proof.

6.3.3 Example Analysis

As an example for the illustration of the analysis framework, consider the little proof problem in propositional logic:

$$(X \wedge Y) \wedge Z \Rightarrow X \wedge (Y \wedge Z)$$

After the proof problem is solved, for example in the PSYCOP framework as modeled in ΩMEGA, the proof can be analyzed in the granularity analysis framework. It is then evaluated according to a particular *Evaluation Schema*, which represents a hypothesis on how a granularity measure can be obtained.

As a part of the granularity framework, there is an automatic transformation procedure from the proof in ΩMEGA to the underlying representation for the granularity analysis framework. In order to connect other proof systems to the analysis framework, a mapping of proofs in these system to the proof representation employed here needs to be provided.

The proof as it is represented in the granularity analysis framework is displayed in the form of a graph in Figure 6.2. It contains eight proof lines, which are labeled with the mathematical statements $(r \wedge s) \wedge t$, $r \wedge s$, t, r, s, $s \wedge t$, $r \wedge (s \wedge t)$, $(((X \wedge Y) \wedge Z) \Rightarrow (X \wedge (Y \wedge Z)))$. There are five deduction rule applications which are labeled with the names of the corresponding deduction rules. Arrows connect

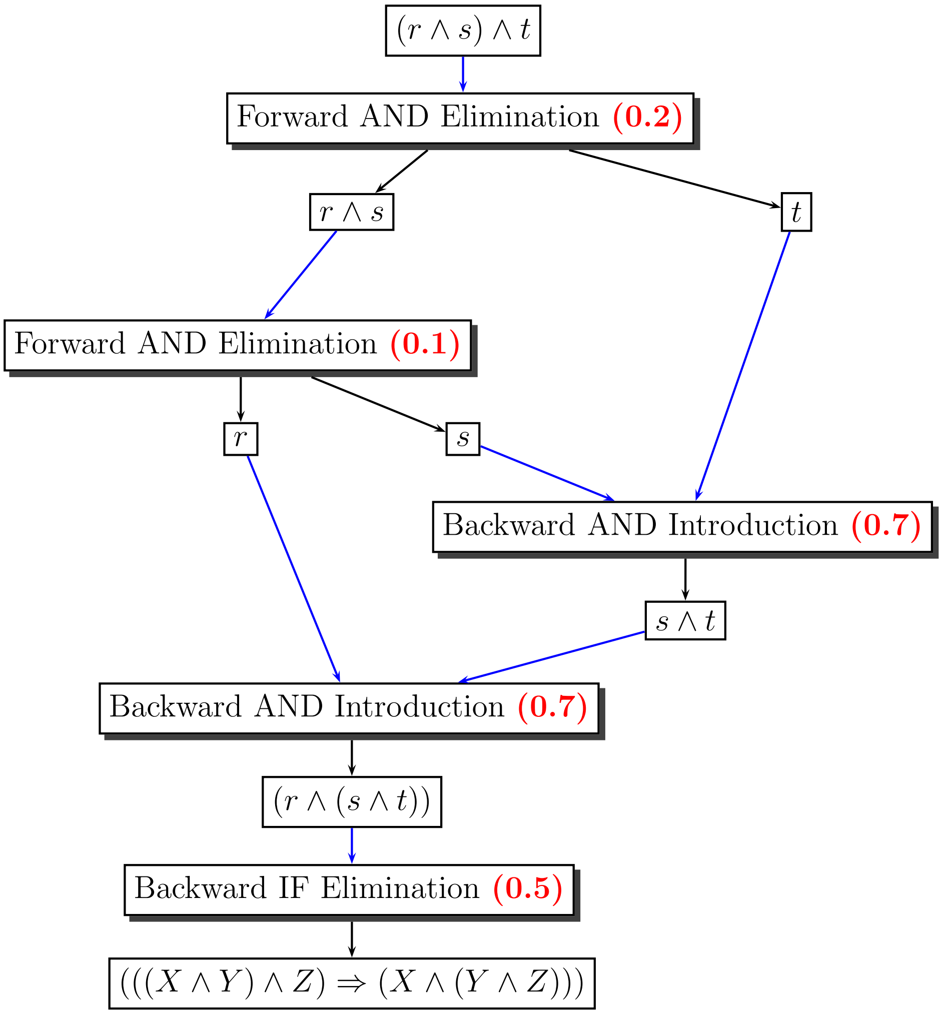

Figure 6.2: Example proof as a graph

deduction rule applications to those proof lines that are justified by them. Another set of arrows connects proof lines to those deduction rule applications that have them as premises. Note that the *Forward AND Elimination* rule in PSYCOP has always two conclusions. If this proof is to be represented as a tree, this can only be done by duplicating some subtrees, as illustrated in Figures 6.2 and 6.3.

$$\dfrac{\dfrac{\dfrac{\dfrac{[((r\wedge s)\wedge t)]^1}{(r\wedge s)}\,\text{Rips-Ande}}{r}\,\text{Rips-Ande} \quad \dfrac{\dfrac{\dfrac{[((r\wedge s)\wedge t)]^1}{(r\wedge s)}\,\text{Rips-Ande}}{s}\,\text{Rips-Ande} \quad \dfrac{[((r\wedge s)\wedge t)]^1}{t}\,\text{Rips-Ande}}{(s\wedge t)}\,\text{Rips-Bckw-Andi}}{(r\wedge(s\wedge t))}\,\text{Rips-Bckw-Andi}}{(((X\wedge Y)\wedge Z)\Rightarrow(X\wedge(Y\wedge Z)))}\,\text{Rips-Bckw-If-Intro}^1$$

Figure 6.3: Example proof in tree form

The set of deduction rules under consideration for the PSYCOP theory is specified as a part of the evaluation schema. For any of the possible rules, an evaluation function can be defined. If no function is defined for a deduction rule, it is replaced by a default evaluation function.

It is now possible to formulate how the different deduction rules are to be evaluated. For example, a hypothetical granularity measure expressed as an evaluation schema is obtained by

- Defining a constant parameter for each application of *Backward AND Introduction* with value 0.7.
- Defining a constant parameter for each application of *Backward IF Introduction* with value 0.5.
- Defining a stateful parameter for each application of *Backward AND Elimination* with an initial value of 0.2, dropping by a factor of 0.5.

As a first step of the evaluation procedure, the parameter values are calculated for each deduction rule application. In Figure 6.2, these are indicated in brackets after the names of the deduction rules. Then these values are combined. For example, the calculation of the sum yields a value of 2.2 for this particular example.

6.4 Modeling of Granularity Hypotheses

The framework described above provides an infrastructure for the analysis of proofs. Since it is generic and does not by itself implement any particular measure for granularity or any other property of proofs, concrete *Evaluation Schemata* have to be provided.

6.4.1 Simple "Addition" Hypothesis

One of the simplest forms to account for the "size" of a proof (fragment) is to count the number of proof steps in a particular calculus.

This supposes that:

- All proof steps count towards granularity in the same way, independent of the qualities of the deduction rule applications.
- The "size" of a proof step is represented by the sum of its constituents (no interactions, and no repetition effects are considered).

Therefore, the accuracy of this hypothesis depends on how well the employed proof calculus represents the "grain size" of the proofs it allows.

The two candidates in this thesis for testing the fit are Gentzen's natural deduction calculus [20] and the PSYCOP theory [40].

The evaluation schema that represents the simple "addition" hypothesis is defined as follows: The deduction rule set $\mathcal{R}$ contains the natural deduction inference rules. We define a constant function f_1 as $f_1(x) = 1$ for all x. A second constant function f_0 is defined as $f_0(x) = 0$ for all arguments x. Then we define the evaluation function assignment γ as $\gamma(\mathrm{x}) = f_1$ if x is a natural deduction rule and $\gamma(\mathrm{x}) = f_0$ if x is any other proof argument.

This will annotate each deduction rule application with 1 if it corresponds to a natural deduction rule, and 0 otherwise. These values are combined with the sum operator.

In the current implementation of the framework, it suffices to specify f_1 only and to indicate a default deduction rule application valuation which constantly evaluates to 0.

6.4.2 Other Hypotheses

Studies on the psychology of logic have shown that deduction rules differ in their cognitive qualities (for example, Byrne and Johnson-Laird [30] empirically determined different levels of difficulty for deductive inferences). In order to formulate a hypothesis where different deduction rules make different contributions towards the evaluation result, the granularity analysis framework allows to specify a collection of different functions that perform a rating for a set of given deduction rules, and which can be sensitive to the arguments and parameters of a deduction rule application.

A hypothesis that is not intended to represent granularity, but the accuracy of human reasoners w.r.t. deductive inferences is presented in [40] and introduced in Section 7.3.1 in the context of Lance Rips' PSYCOP theory. It is mentioned here to demonstrate the framework's capability to represent such hypotheses in it. According to this hypothesis, the deduction rule applications are associated with a value that represents the probability that a human reasoner in one of Rips's studies is able to apply a particular deduction rule correctly. Rips supposes that the probability to find a particular proof is the product of the single probability values. The performance data from the subjects in Rips's experiments was then used to estimate the probability values for a number of inference rules. Lance Rips observes that these values appear to be stable across different experiments.

This hypothesis is modeled in the granularity evaluation framework by assigning to each deduction rule an evaluation function that yields the corresponding probability value established by Lance Rips in [40]. The resulting probabilities are combined with the product operator.

Even though this hypothesis is not about granularity, it fits into the framework. This example case demonstrates the formulation of hypotheses which treat each deduction rule application separately. This way, the analysis framework allows hypotheses that go beyond simple counting.

6.5 Discussion

We have presented a framework for the qualitative evaluation of proofs. It includes and operates on a graph-like proof representation, where the applications of deduction rules are treated as reified concepts. Different hypotheses on how to obtain qualitative measures from a proof can be declared in the form of *Evaluation Schemata*. The evaluation mechanism determines parameter values for each deduction rule application - according to such a hypothesis - and combines them to obtain a qualitative measure.

The most demanding task for the qualitative analysis of a proof is to find a suitable hypothesis for measurement. The analysis framework provides the infrastructure to mechanize such a hypothesis, but it does not help in finding a hypothesis in the first place. Several *Evaluation Schemata* have been implemented in this thesis, but many of them are very simple (like the simple "addition" hypothesis in Section 6.4.1) and do not require the advanced features of the framework. Analyzing a proof with a less simplistic hypothesis is not difficult in the framework once such a hypothesis has been found - the challenge is the process of obtaining and verifying it. In this thesis, a simple hypothesis is tested, which claims that the number of justifying proof steps for a mathematical statement identifies that statement's status w.r.t. granularity. This is done by using the granularity analysis on proofs in Gentzen's natural deduction calculus [20] and the PSYCOP theory [40], which is presented in the next chapter.

Chapter 7

PSYCOP - A Cognitive Theory on Logic

The "Psychology of Proof" is a psychological theory on human deductive reasoning by Lance J. Rips presented in [40]. The theory consists of a formal account of the processes underlying human reasoning. It is based on the so-called *Deduction System Hypothesis*, a psychological paradigm that claims that humans - even if untrained in formal logics - possess a set of abstract formal rules that is central to deductive reasoning. The work of Lance J. Rips shows how a human-oriented calculus in the tradition of Gentzen's *natural deduction* calculus [20] can be designed with the aim of psychological plausibility. The resulting deduction system named PSYCOP (after "PSYChology Of Proof") has been implemented in PROLOG. In the course of designing and testing the theory, a considerable number of empirical studies have been carried out which - as Rips concludes - speak in favor of the theory.

7.1 Characterization

The *Deduction System Hypothesis* explains logical reasoning abilities in humans in terms of a general-purpose deduction rule system, which is supposed to account for deductive problem solving as well as for everyday reasoning. The fundamental processes that underlie this deductive capability are considered as being innate to humans, who may use them without any prior training in formal logics.

An alternative to rule-based approaches like the *Deduction System Hypothesis* is the model-theoretic approach, among which the *Mental Model* theory proposed by Philip Johnson-Laird is probably the most prominent to date. The *Mental Model* theory advocates that human reasoners develop mental representations of possible state of affairs for a given logical problem. The theory postulates that limitations of the cognitive resources of the human reasoners allow them to only consider a partially developed mental model for a given logical problem, which explains systematic reasoning errors that human reasoners commit in psychological experiments. An introduction to *Mental Models* in the wider context of research on deductive reasoning is provided by Johnson-Laird's article [29].

Even among rule-based theories there are fundamental conceptual differences. Unlike PSYCOP, where a particular set of deduction rules is designated to explain human reasoning capabilities on a variety of tasks in logic and problem solving, other rule-based theories argue for domain-specific rules, like the *Pragmatic Reasoning Schemas* by Patricia Cheng and Keith Holyoak presented in [15].

There is a yet unresolved dispute between the proponents of the rivaling theories. Thus it is unclear whether the *Deduction System Hypothesis* provides a better explanation for those processes of the mind that carry out reasoning tasks. But among the different theories, the *Deduction System Hypothesis* has the advantage that rule-based deduction systems have been studied intensively in the *theorem proving* communities. This is why rule-based deduction system architectures like PSYCOP can be characterized in terms of properties like decidability or their expressiveness, and they can be implemented as computer programs and run as simulations (as demonstrated by Lance Rips in [40], and also in this thesis).

7.1.1 Inference Rules in PSYCOP

The starting point for the formalization of the inference rules in the PSYCOP theory is Gentzen's natural deduction calculus NK [20]. In the following paragraphs, the reader is assumed to be familiar with Gentzen's natural deduction calculus. Gentzen's rule set is modified with the aim of psychological plausibility. First, the rules for propositional logic are elaborated, then, in a second step, rules for quantified formulae are introduced that work on implicit "quantifier-free" (i.e. implicitly quantified) representations for first-order predicate calculus. The complete set of rules in PSYCOP is reproduced in Appendix D.

The Propositional Case The deduction rules for propositional logic in PSYCOP contain the rules from Gentzen's NK for *AND Introduction*, *AND Elimination*, *OR Introduction*, *OR Elimination*, *IF Introduction*, *IF Elimination* and *NOT Introduction* with some modifications, e.g. the two "deduction rule schemata" that Gentzen proposes for *AND Introduction* in [20] are collapsed into one single *AND Introduction* rule in PSYCOP. The *Tertium non datur* rule in [20] is represented in the form of *Double Negation Elimination* in the PSYCOP rule set.

The set of these deduction rules is enlarged by further rules that, according to Rips, "the individual recognizes as intuitively correct". These are rules for

$$\frac{\begin{array}{l}(P \wedge Q) \rightarrow R \\ P \\ Q\end{array}}{R}$$

Conjunctive Modus Ponens

$$\frac{\begin{array}{l}P \vee Q \\ P \rightarrow R \\ Q \rightarrow R\end{array}}{R}$$

Forward Dilemma

$$\frac{\neg(P \wedge Q) \quad P}{\neg Q} \qquad \frac{\neg(Q \wedge P) \quad P}{\neg Q}$$

Conjunctive Syllogism

$$\frac{P \vee Q \quad \neg Q}{P} \qquad \frac{P \vee Q \quad \neg P}{Q}$$

Disjunctive Syllogism

$$\frac{(P \vee Q) \rightarrow R \quad P}{R} \qquad \frac{(P \vee Q) \rightarrow R \quad Q}{R}$$

Disjunctive Modus Ponens

$$\frac{\neg(P \wedge Q)}{\neg P \vee \neg Q} \qquad \frac{\neg(P \vee Q)}{\neg P \wedge \neg Q}$$

De Morgan

Figure 7.1: Intuitively correct propositional PSYCOP rules. Source: Lance J. Rips. The psychology of proof: deductive reasoning in human thinking. MIT Press, Cambridge, MA, 1994

Conjunctive Modus Ponens, Forward Dilemma, Conjunctive Syllogism, Disjunctive Syllogism, Disjunctive Modus Ponens, and two *DeMorgan rules*, as shown in Figure 7.1. Furthermore, the propositional PSYCOP rules include a *NOT Elimination* rule which does not directly correspond to any of the rules in Gentzen's NK, but which can be justified with Gentzen's *NOT Introduction* and *Tertium non datur* rules.

$$\frac{A \quad \neg A}{\perp}$$

Figure 7.2: NOT Elimination

$$\frac{\perp}{D}$$

Figure 7.3: $\perp$ Elimination

Gentzen's deduction rules for *NOT Elimination* (also named *reductio ad absurdum*[1] in [20], and presented in Figure 7.2) and the unnamed deduction rule[2] in Figure 7.3 have no immediate counterpart in PSYCOP.

Rips distinguishes between *self-constraining* and *self-promoting rules*, the former being those rules that are limited in the number of new sentences they can produce from a given set of assertions, and the latter producing a potentially infinite number of new sentences from one assertion set. Self-constraining rules, like *AND Elimination* are only used in forward direction, whereas self-promoting rules, like *AND Introduction* are used only in backward direction to constrain the number of forthcoming inferences. The full rule set is reproduced from [40] in Appendices D.1 (forward rules) and D.2 (backward rules).

$$\frac{A}{\neg A \Rightarrow B}$$

Figure 7.4: Paradox of implication

The resulting rule system is incomplete with respect to classical propositional logic. Lance Rips points out that one of the paradoxes of implication, which is shown in Figure 7.4, cannot be shown with the PSYCOP rule system for prepositional logic. He develops an extended rule system with two more rules (reproduced

[1] Here, $\perp$ denotes falsehood.

[2] Here, D denotes any propositional formula.

in Appendix D.3) which he calls PSYCOP+ and shows that it is complete for propositional logic. But Rips prefers the rule system of PSYCOP for his theory on deduction, because he considers the paradox of implication in Figure 7.4 as an unintuitive argument. This is in line with the intention to design a calculus that reflects intuitive reasoning without enforcing proof-theoretic completeness.

The Case of Classical Predicate Logic The rule set for propositional logic is further modified to accommodate reasoning with variables and quantifiers. Quantification is represented "implicitly", similarly to the approaches by Skolem [45], Wang [47], Bledsoe [10] and Murray [37].

Universal quantification is expressed by using variables (without explicit representation of the universal quantifier), whereas existential quantification is expressed using Skolem functions or Skolem constants, with "temporary names" standing for particular individuals. The arguments of the Skolem functions are represented in the form of subscripts that are attached to the temporary names.

Rips distinguishes between two classes of temporary names: "We must also take the further precaution of differentiating the temporary names that appear in the premises from those that appear in the conclusion, using carets over all temporary names in the premises as we did earlier. In a sense, temporary names in the premises behave like permanent names, whereas temporary names in goals behave more like variables."(cf [40], p. 190).

$$\frac{\begin{array}{c}F(x) \rightarrow G(x)\\ G(y) \rightarrow H(y)\end{array}}{F(z) \rightarrow H(z)} \qquad \frac{\begin{array}{c}F(x) \rightarrow G(x)\\ \neg(G(y) \wedge H(y))\end{array}}{\neg(F(z) \wedge H(z))} \qquad \frac{\neg(F(y) \wedge G(y))}{\neg(G(x) \wedge F(x))}$$

Transitivity Exclusivity Conversion

Figure 7.5: Additional forward PSYCOP rules

The inference rules for quantified sentences directly move from one quantified expression to another. Rips refers to evidence by Braine & Rumain [11] that human reasoners seem to be able to do so "in many situations". The forward rules for propositional logic are carried over to the first-order case. Three additional forward rules are added, namely transitivity, exclusivity, and conversion as shown in Figure 7.5. The backward rules are modified: for each backward rule in propositional calculus there is a new backward rule that works on quantified sentences. The rules are expressed in a procedural form, most of which require auxiliary procedures for matching, "argument reversal", and "subscript adjustment". The "argument reversal", and "subscript adjustment" procedures are necessary for dealing with negation and conditionals. *Backward Modes Ponens* is an example for a backward rule which is expressed in the form of an algorithm in [40]. It is displayed in Figure 7.6.

Backward IF Elimination (Modus Ponens)

(a) Set R to the current goal and set D to its domain.
(b) If R can be matched to R' for some sentence IF P' THEN R' that holds in D,
(c) then go to step (e).
(d) Else, return failure.
(e) If P' and R' share variables and one or more names or variables in R matched these variables,
(f) then set P to the result of substituting those names or variables for the corresponding variables in P'. Label the substituting arguments the matched arguments of P and the residual arguments the unmatched arguments of P.
(g) Else, set P to P'. Label all its arguments as unmatched.
(h) Apply Argument Reversal to unmatched arguments of P.
(i) Apply Subscript Adjustment to output of step (h). Call the result P^*.
(j) If D does not yet contain the subgoal P^* or a notational variant,
(k) then add the subgoal of proving P^* in D to the list of subgoals.

Figure 7.6: Modus ponens (backward) in PSYCOP. The procedure for Backward IF Elimination applies to an open proof goal R in quantifier-free form within a PSYCOP proof. The domain D of the proof goal R is defined as the set of proof lines in the current proof that share the same set of assumptions that are made for R. If the specified conditions are satisfied, the procedure changes the proof structure by inserting a new subgoal. Source: Lance J. Rips. The psychology of proof: deductive reasoning in human thinking. MIT Press, Cambridge, MA, 1994, p. 197.

The *Backward Modes Ponens* rule makes use of the matching mechanism which is based on four matching rules. These matching rules cover the following cases:

1. A(n) (all-quantified) variable x in a subgoal may match another (all-quantified) variable y in a sentence.

2. A temporary name in a subgoal can match a temporary name or a constant in a sentence.
3. A constant in a subgoal can match a variable in a sentence.
4. A temporary name in a subgoal can match a variable in a sentence.

The actual rules involve complex conditions and actions, which are not explicated here. As an example, a full matching rule is presented in Figure 7.7.

Matching 4 (temporary names in subgoal to variables in assertion)

Conditions:
(a) $P(x_1, ..., x_k)$ is a target formula and $P'(t)$ is an isomorphic subgoal.
(b) $x_1, ..., x_k$ are variables and t is a temporary name.
(c) x_1 or x_2 or $...x_k$ appears in non-subscript positions in $P(x_1, ..., x_k)$ at each position that t occupies in $P'(t)$.
(d) if a temporary name t' appears in $P'(t)$ in the same position that an x_i-subscripted temporary name t_{x_i} appears in $P(x_1, ..., x_k)$, then any additional tokens of t' also appear in positions occupied by t_{x_i}.

Action:
Add the subgoal of proving $P'(x_1, ..., x_k)$ to the list of subgoals, where $P'(x_1, ..., x_k)$ is the result of:
(a) substituting x_i for t wherever t occupies the same position in $P'(t)$ than x_i occupies in $P(x_1, ..., x_k)$.
(b) substituting t'' for t' in $P'(t)$ wherever t' occurs in the same position as an x_i-subscripted temporary name in $P(x_1, ..., x_k)$ and where t'' is a new temporary name whose subscripts consist of x_i and all subscripts of t'.

Figure 7.7: Matching rule 4. Source: Lance J. Rips. The psychology of proof: deductive reasoning in human thinking. MIT Press, Cambridge, MA, 1994, p. 192.

7.1.2 Heuristics

In addition to providing a psychologically motivated calculus, Rips specifies heuristics that hypothetically apply to proof search by human reasoners. It is postulated that applying an inference rule in backward direction is "harder" than using it in forward direction. According to Rips, certain rules are only used in backward direction, in particular those rules that involve sub-domains, such as *IF Introduction*, *NOT Introduction*, *NOT Elimination* and *OR Elimination*. This is in line with the distinction by Huang [26] between *structural* (assumption-introducing) and *non-structural* natural deduction rules.

When attempting a proof for a logical argument, PSYCOP proceeds as follows:

- Forward rules are applied until no new conclusion is forthcoming.
- Then backward rules are attempted in the order of supposed difficulty. If a new subgoal is introduced this way, the forward rules are re-run.
- In case the forward rules produce no new conclusions, the backward rules are applied in a depth-first manner to the open subgoals.

7.2 Formal Properties

The PSYCOP theory is proven to be sound, but it is incomplete w.r.t. classical predicate logic. For example, the argument

$$\frac{\neg(P(x) \Rightarrow Q(x))}{P(a)}$$

cannot be proven in PSYCOP. This is not due to the choice of a quantifier-free representation, since there exist quantifier-free systems that are complete (cf. [45], [39]). Rather, the aim of Rips is to exclude inference rules that are unintuitive to people untutored in logic, even though these rules could potentially contribute to the completeness of the system. Rips rejects the existing complete calculi by saying that "what such systems gain in simplicity they lose in psychological fidelity, since they don't correctly reflect human proof strategies" (cf. [40], p. 214).

7.3 Empirical Studies

The plausibility of the theory has been tested by Lance Rips in two independent studies. In both of these, logical arguments were presented to subjects who had to decide whether the conclusion such an the argument "necessarily follows".

The first study concentrated on testing the PSYCOP theory for propositional logic only, whereas the second was concerned with logical argumentation in classical predicate calculus involving quantifiers.

7.3.1 The Propositional Case

The logical arguments presented to the subjects were either logically correct (in a classical natural deduction system), in which case they could be solved with a maximum of three rule applications of inference rules in PSYCOP, or the conclusion didn't follow - these arguments were constructed by combining arbitrary premises and conclusions. The dependent variable in the experiment was the percentage of "necessarily follows" and "doesn't follow" answers for each problem. This data was then be used to estimate parameters for PSYCOP which were used for fitting the theory to the empirical data.

The subjects in the study were students and non-students. According to Rips, none of these had taken a course in formal logics. The logical problems were presented without using standard notation for logics (symbols for connectives and quantifiers such as $\neg$, $\vee$, $\wedge$, $\forall$, $\exists$), but framed in natural written language. Thus, as an example, the logical problem

$$\frac{(p \vee r) \to q}{(p \vee q) \to q}$$

was presented to the subjects in the form

$$\frac{\text{If Judy is in Albany or Barbara is in Detroit, then Janice is in Los Angeles}}{\text{If Judy is in Albany or Janice is in Los Angeles, then Janice is in Los Angeles}}$$

The problems were presented to the subjects as a list in random order.

Rips assumes that the subjects will respond to a problem by "necessary follows" if they are able to construct a mental proof. In case they cannot construct such a proof, they will simply guess. If they fail to construct a proof, this might be due to retrieval difficulties, incorrect application of inference rules, failure to recognize that a particular rule is applicable, or "other factors" (cf. [40], p. 153). Therefore, each inference rule is associated with a particular probability that it will be "available" to a subject during a particular experiment trial, i.e. that the subject can retrieve the rule from memory and apply it appropriately. The probability for finding a particular proof for the given problem is assumed to be the product of the probabilities of the inference rules that constitute the proof. Furthermore, the probability for responding "necessarily true" to a given problem is the sum of the probabilities associated with the different proof variants that can be found for this problem, plus the probability of correctly guessing the right answer without producing a proof. The resulting equations yield availability parameters for the different inference rules when fitted to the relative frequencies determined in the experiment. Even though the parameters for forward and backward versions of a rule were collapsed into one parameter, there remained 21 degrees of freedom for fitting the data. These parameters where then used to replay the experiment with PSYCOP.

7.3.2 The Case of Classical Predicate Calculus

The PSYCOP theory for reasoning with quantifiers was evaluated using a similar experiment as described above for propositional logic. The subjects, 20 Stanford undergraduates not trained in formal logics, were given a booklet with problems from introductory textbooks on predicate logic that could be proven by PSYCOP. These problems were mixed with invalid arguments. Again, these problems were framed in natural language as sentences about members of a club. Thus, the argument

$$\frac{\text{IF F(x) THEN G(a)}}{\text{IF F(b) THEN G(b)}}$$

was presented to the subjects in the form

$$\frac{\text{There's a person A such that for any person B: if B is cheerful then A has glasses.}}{\text{There's a person C such that: if C is cheerful then C has glasses.}}$$

The subjects were instructed that the different letters in the arguments not necessarily designate different persons. Again, the subjects had to decide whether the conclusion "follows" or "does not follow". As it was the case in the corresponding experiment in propositional logic, the observed responses were used for estimating parameters for the availabilities for a subset of PSYCOP's inference rules.

7.3.3 Results

The empirical data from the experiments presented above was used for fitting it with the parameters of PSYCOP's model. The data from running PSYCOP on the problems from the both experiments revealed a correlation of 0.93 and 0.806 respectively with the empirical data.

With respect to the estimated parameters, Rips observes that: "the easiest rules are those that seem obviously correct". In line with other studies, the experiments provide further evidence that *NOT Introduction* is difficult for subjects without extensive mathematics training.

Most interestingly, the estimates for inference rules in the propositional case and their corresponding counterpart in the experiment for classical predicate logic seem to reveal a high degree of correspondence. Rips thus postulates that the results for the parameters are "stable" across the two experiments, and thus support the idea of a quantifier-free inference system.

Rips's analysis of other surface measures for the complexity of a logical argument yields negative results. Neither the number of premises, nor the number of atomic sentences in a logical argument appear to be significantly correlated to the difficulty of such an argument.

7.3.4 Further Empirical Studies

Rips carried out further studies, some of which aim at explaining previous findings in the light of the PSYCOP theory, such as an experiment on *logical syllogisms*. But he also provides a study on *proof comprehension*, by having PSYCOP automatically fill in the missing arguments in a "gappy" proof (thus predicting reaction times for human reasoners when trying to understand the proof). A study on *memory for proof* shows that a sentence from a proof is remembered less well when it occurs in a sub-domain rather than a in super-domain. The notion of a domain identifies all sentences in a proof in a particular proof context, i.e. under the same assumptions. A sub-domain is derived from its super-domain by adding one further assumption, thus making the sub-domain more "hypothetical". Another experiment is concerned with the four matching rules proposed by Rips. He shows that not the total number of matching rule applications, but the number of different rules needed for solving a matching problem has an influence on the reaction times of human reasoners.

7.4 A Realization of PSYCOP Rules in ΩMEGA

The deduction rules in the PSYCOP theory are in the tradition of Gentzen's *natural deduction* calculi [20]. Therefore, PSYCOP can be understood as an adaption of classical natural deduction.

This section describes how a substantial part of the PSYCOP rule system was implemented in the ΩMEGA system [44], a software environment for interactive and automated theorem proving. Since a higher-order variant of Gentzen's natural deduction calculus is already represented in ΩMEGA, this work allows to compare these two natural deduction calculi in one unified framework. This framework was then used for the comparative study on granularity presented in Chapter 8.

The implementation of PSYCOP in ΩMEGA includes all rules for propositional logic, mechanisms for working with quantifier-free representations like the matching mechanism, the "subscript adjustment" and the "argument reversal" procedures, and those rules for first-order logic that are needed to prove the mathematical problems developed for the experiment in Chapter 3. This section shortly presents the ΩMEGA system, reviews the algorithmic construction of the PSYCOP theory and highlights those aspects that deserve special attention when moving the elements of PSYCOP to the general ΩMEGA framework.

7.4.1 A Short Introduction to ΩMEGA

ΩMEGA is a mathematical assistant tool with a modular structure. At the core of the system, the *PDS* (Proof Data Structure) provides the facility to store and develop proofs at different levels of detail. The system is based on a higher-order natural deduction variant of Church's simple λ-calculus [16]. Mathematical problems are specified in a language called *POST* (partial functions order sorted type theory), which allows variables and constants, logical connectives, quantifiers, lambda abstraction and application.

Grouped around the central data-structure *PDS*, different modules offer specific services like automated theorem proving (in the form of proof planning), agent-based support for interactive theorem proving (the so-called Ω-ants [8]) or proof verbalization (the *P.rex* system [17]). The idea behind the ΩMEGA system is to provide a platform to integrate a variety of assistance tools. External theorem provers can be called from within ΩMEGA, and their results are made available within the ΩMEGA system.

Besides a higher-order version of Gentzen's natural deduction calculus NK [20], the ΩMEGA system offers a definition expansion mechanism, which replaces defined constants by their definitions. These definitions are formulated in the *POST* language. Together with the simple syntactic replacement of the constant by the body of the definition, the mechanism performs β-reduction (in the spirit of Church's simple λ-calculus [16]).

New deduction rules are added to the ΩMEGA system in the form of so-called "tactics". Tactics enable the representation of a more abstract level of deduction rules with respect to a ground calculus level, since a tactic can express the result of a sequence of other - more fine-grained - inference steps. Replacing an instance of a tactic in proof by the corresponding proof steps on the lower abstraction level is called tactic expansion. A tactic usually represents several application directions of a deduction rule. For example, the *Modus Ponens* deduction rule (as in [20]) can be used in forward direction (deriving a new conclusion) as well as in backward direction (introduce a new subgoal to justify the conclusion).

7.4.2 Quantifier Free Representations in ΩMEGA

The PSYCOP rule system operates on formulae in quantifier-free form, i.e. quantification is represented implicitly. Lance Rips mentions that this form is also called "Skolem function form" and "functional normal form" and refers to [39], [22]. Lance Rips points out that "temporary names in this context are sometimes called *Skolem constants* (if unsubscripted) or *Skolem functions* (if subscripted)" (cf. [40], p.92). He explains that "temporary names stand for particular individuals depending on the values of any variable that subscript them" (cf. [40], p.93).

This quantifier-free formula representation in PSYCOP (as introduced in Section 7.1.1) has been carried over to the ΩMEGA system with one difference in the representation of "temporary names". Lance Rips considers a temporary name as an identifier for an object and attaches variables to it as its subscripts. These subscripts represent the universally-quantified variables upon which the object depends. Another variant, which is more natural in ΩMEGA, is to represent a temporary name as a Skolem function (respectively Skolem constant) applied to the corresponding universally-quantified variables. This representation is used for representing quantifier-free formulae in ΩMEGA as part of this thesis.

Lance Rips distinguishes between temporary names that appear in the assumptions of a proof problem and the ones that only appear in other parts of the proof. In his proof examples, the former are marked with a caret (a hat). Since the variables in ΩMEGA do not wear hats, this role is taken by a predicate function determining whether a particular temporary name has an appearance in the premises of the proof problem at hand.

The quantifier-free form is obtained from a formula in classical logic with quantifiers in a sequence of steps:

(a) Assign a unique name to each quantified variable.

(b) Put the formula into prenex form.

(c) Introduce a temporary name for each existentially bound variable. All universally quantified variables in whose scope an existentially bound variable occurs become arguments of the corresponding Skolem term, or "subscripts" in PSYCOP terminology.

(d) Drop universal quantifiers and "free" the bound variables

A more detailed account of the procedure can be found in the section "Quantifiers" in chapter 3 of [40]. As part of this thesis, the procedure has been implemented within the ΩMEGA framework, so quantifier-free formulae can be created automatically from the representation in Church's type theory which is standard in ΩMEGA. So proofs in Gentzen's natural deduction calculus [20] as well as PSYCOP proofs can be performed starting with the same problem representation.

Another concept that appears in [40] is that of a "domain", which is defined by Rips as "a designated set of lines of the proof that are associated with a supposition" (cf [40], p.39). He illustrates the notion of a domain via an example.

In the following, an attempt for a formal definition of a "domain" in PSYCOP is given:

Assume a given natural deduction proof P that consists a set of proof lines L. For each proof line $l \in L$ there is a set of proof lines that represent the suppositions under which the proof line l is deduced (possibly empty). This provides a mapping $\tau : L \Rightarrow \mathcal{P}(L)$ that assigns to each proof line a set of suppositions (which correspond to the so-called "supports" in the ΩMEGA system). A domain is defined as a set of proof lines in the proof P under the same set of suppositions S, formally: $D(S) = \{l \in L | \tau(l) = S\}$. Often, Rips refers to the domain of a proof line. This refers to the domain defined by the suppositions for that proof line. In the PSYCOP system, a sub-domain D' is created from a given domain D by adding a further formula to the set of suppositions for D. As an example, given a proof with a proof goal of the form $P \Rightarrow R$, *Backward IF Introduction* creates a new goal proof R', which has to be proven under the suppositions for $P \Rightarrow R$ (the domain of $P \Rightarrow R$) and the new hypothesis P', which together form a new sub-domain. Formally, a domain D' (defined by a set of hypotheses S') is a sub-domain of a domain D (defined by a set of hypotheses S), if $S' \supseteq S$, i.e. S' is actually a superset of S(!).

In the PSYCOP adaption for ΩMEGA, the domain of a particular proof line is determined by considering all proof lines as candidates and filtering out those that do not share the same set of supports with the given proof line.

7.4.3 PSYCOP Deduction Rules in ΩMEGA

In the PSYCOP theory, a considerable number of deduction rules (namely the backward rules for first-order logic) are given in a procedural form, i.e. they are presented in the form of step-by-step algorithms with conditions and actions. In the book [40], these algorithms are presented as text paragraphs in natural language.

Several backward rules in PSYCOP that are indispensable for first-order logic require sub-procedures like the matching sub-procedure, an "argument reversal" procedure and a "subscript adjustment" procedure. These have been reproduced in ΩMEGA as part of this thesis. The matching mechanism and its representation in ΩMEGA are described in Section 7.4.5.

Differently from standard natural deduction calculi like Gentzen's NK [20], some of the rules in PSYCOP trigger the application of other rules (and possibly an instance of the same rule again), then collect a result from the rule applications and continue with their own actions accordingly. Whereas in the case of conventional rule systems, the deduction rules are applied one after the other, the PSYCOP system can pass the flow of control back and forth between different rule applications. Therefore it can be the case that a particular application of a deduction rule is "pending" while others are being applied. Examples for such "rules" are given in the section 7.4.5, together with observations and problems that arose while implementing these rules in ΩMEGA, as well as their solutions.

The most elaborate example for a proof with quantified variables in the PSYCOP framework provided in [40] is presented in Section 7.4.6. This proof has been reproduced in the ΩMEGA framework for demonstration purposes.

7.4.4 Rules for Propositional Logic

The PSYCOP deduction rules for propositional calculus are provided in [40] (pages 113f, 116ff). They are very similar to the natural deduction rules that are available by default in ΩMEGA. The PSYCOP rule set for propositional logic was implemented as a set of 20 "tactics" in ΩMEGA, including the rules that Lance Rips proposes for obtaining a complete (but not psychologically plausible) system for propositional logic. Those rules in PSYCOP that have a backward and a forward version were implemented as two application directions of the same tactic, thus the set of tactics accounts for all of the 24 rules presented in [40] plus the two extra rules for completeness.

7.4.5 Rules for First-Order Logic

As mentioned in Section 7.1.1, formulae in PSYCOP are represented in quantifier-free form. Thus, all names and variables in a formula represent quantification explicitly. The PSYCOP theory postulates that from a formula F in quantifier-free form another formula F' can be deduced, provided the following conditions hold:

- Formula F and formula F' are isomorphic.
- All variables and names in formula F' can be matched to the variables and names in the corresponding positions in F.

A formula is said to be isomorphic to another formula if they share the same syntactic structure and only differ in their variables and temporary names.

Matching Rules

Matching of variables and names is provided by four "matching rules", which are presented in the form of procedures in [40]. As an example, the rule named "Matching Rule 4" is presented in Figure 7.7. Given a quantifier-free formula F as an assertion, and a quantifier-free formula F' as a proof goal to be proven under F, the matching mechanism outlined in [40] attempts to successively match the names and variables in the proof goal to those in the assertion. The procedure stops with success if both formulae contain equal names and variables in the corresponding positions. This procedure terminates for formulae of finite length, since each variable or temporary name is only tried once.

The matching mechanism is used as a part of the *Backward IF Elimination*, *Backward AND Elimination*, *Backward Double Negation Elimination*, *Backward*

Disjunctive Syllogism, *Backward Disjunctive Modus Ponens*, *Backward Conjunctive Syllogism* and *Backward DeMorgan* rules. Furthermore, the matching mechanism is invoked after any application of another inference rule to determine whether the formula or formulae that are newly introduced into the proof can be matched with any of the previously existing ones. An example for the use of matching both as part of a backward inference rule and as a rule on its own is given in Section 7.4.6.

In the course of the matching procedure, the variables and names in a proof goal are successively replaced by the variables and names in the assertion they are matched to. These replacements are not only of interest for the matching process itself, but they provide input for the actions of the inference rules. They cannot, though, be represented by substitutions, since substitutions are usually understood as mappings that relate all instances of a variable in a formula to another expression that substitutes it. On the other hand, the PSYCOP matching rules require a selective replacement of some appearances of the same identifier in the same formula.

An example for such a scenario is quickly at hand. Assume that the following formula is given as an assertion in a proof problem:

$$\forall x(\forall y(\forall z(R(x,y) \wedge R(y,z) \Rightarrow R(x,z)))),$$

or in quantifier-free form:

$$R(x,y) \wedge R(y,z) \Rightarrow R(x,z).$$

Further assume that a second assertion

$$\exists a(\exists b(R(a,b))), \text{ in quantifier-free form: } R(a,b)$$

is given, and that the goal of the proof is

$$\exists c(R(c,c)), \text{ in quantifier-free form: } R(c,c), \text{ respectively.}$$

This goal can be proven in PSYCOP with *Backward IF Elimination*. *Backward IF Elimination* requires the proof goal $R(c,c)$ to be matched against the succedent of the implication $R(x,y) \wedge R(y,z) \Rightarrow R(x,z)$. According to Matching Rule 4 displayed in Figure 7.7, this leads to the temporary name c being replaced by x in its first position in $R(c,c)$ and by z in its second position. This selective form of "replacement" is not in line with the general parsimonious concept of a substitution. Because of this, the replacements in the PSYCOP theory required specialized procedures for the ΩMEGA framework .

Backward Inference Rules

The backward rules for logic with quantifiers are presented in a procedural form in [40]. In their original formulation, there are several rules, namely *Backward NOT Elimination*, *Backward NOT Introduction*, *Backward AND Introduction*, *Backward OR Elimination*, *Backward OR Introduction* and *Backward Disjunctive Modus Ponens* which require that their execution is stopped while other rule applications are being executed in order to collect results from these.

Backward AND Introduction

(a) Set D to domain of current goal
(b) If current goal is of the form P AND Q,
(c) and D does not yet contain the subgoal P
(d) then add the subgoal of proving P in D to the list of subgoals.
(e) If the subgoal in (d) succeeds,
(f) and P is matched to P',
and P and Q share temporary names,
(g) then set Q' to the result of substituting in Q any names that were matched to those temporary names.
(h) Else, set Q' to Q.
(i) If D does not yet contain the subgoal Q',
(j) then add the subgoal of proving Q' in D to the list of subgoals.

Figure 7.8: Backward AND Introduction. Source: Lance J. Rips. The psychology of proof: deductive reasoning in human thinking. MIT Press, Cambridge, MA, 1994, p. 198.

The *Backward AND Introduction* rule is crucial for the analysis presented in Chapter 8 of this thesis and was thus implemented in ΩMEGA. Here it will serve as an example to illustrate the challenge of representing such a rule in ΩMEGA. Its original formulation is displayed in Figure 7.8.

The *Backward AND Introduction* rule applies to an open proof goal in quantifier-free representation of the form $P\ AND\ Q$ and reduces it to the two subgoals P and Q', where the former is obtained from the right conjunct Q. But the two new proof goals P and Q' are not introduced at once. An additional condition is that the proof of Q' is only attempted if the proof of P succeeds (which requires the execution of *Backward AND Introduction* to be suspended until a result for P is obtained). Furthermore, the new subgoal Q' depends on the results of the matchings applied to P in the course of its proof, in case Q shares temporary names with P. The necessity for constraining the temporary names in the subgoals introduced by *Backward AND Introduction* is demonstrated by Rips with an example (cf. [40], page 188). As part of the implementation work, this example has been reproduced in the ΩMEGA system to demonstrate the employed mechanisms.

$$\frac{\begin{array}{l}\text{IF Square-Block(x) THEN Green-Block(x)}\\ \text{Big-Block(â) AND Square-Block(â).}\end{array}}{\text{Big-block(b) AND Green-block(b)}}$$

Figure 7.9: Example syllogism

The proof argument is a syllogism displayed in Figure 7.9. The corresponding proof, performed in the ΩMEGA framework with the PSYCOP deduction rules, is presented in Figure 7.10.

The original procedure for the *Backward AND Introduction* rule proceeds in two steps for proving the proof goal *Big-Block(b) AND Green-Block(b)*. It introduces *Big-Block(b)* as a new subgoal (cf. step (d) in Figure 7.8) and the proof search is continued. When the proof for *Big-Block(b)* has been found, the procedure becomes active again (cf. step (e) in Figure 7.8) and introduces *Green-Block(â)* as a second subgoal. The reason is that in the course of the proof of *Big-Block(b)*, the temporary name b was matched to $\hat{a}$, and b is shared between both conjuncts in *Big-Block(b) AND Green-Block(b)*.

Unlike in PSYCOP, in the ΩMEGA system a rule application is expected to terminate before the next rule is applied. In this sense, the rules are "atomic" since they are never carried out partially, but either completely or not at all. Therefore, another solution to carry over the matching from one of the conjuncts in the *Backward AND Introduction* rule to the other was needed to incorporate *Backward AND Introduction* into ΩMEGA.

Differently from the original PSYCOP proof mechanism, the counterpart of *Backward AND Introduction* in ΩMEGA introduces the intermediate goals *Big-Block(b)* and *Green-Block(b)* at the same time in the proof example in Figure 7.10. After matching *Big-Block(b)* to *Big-Block(â)*, the matching mechanism memorizes via a constraint that the temporary name b in *Green-Block(b)* can only be matched to $\hat{a}$. Rips demonstrates that if the second subgoal *Green-Block(b)* is not constrained, the conclusion could wrongly be shown if some block is "green" and some block is "big" even if these are not the same. This is because in quantifier-free form, proving *Big-Block(b) AND Green-Block(b)* is not equivalent to proving the two formulae *Big-Block(b)* and *Green-Block(b)*, since b in the first formula now potentially represents a different individual than b in the second formula.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(\text{Big-Block}(\hat{a})\wedge \text{Square-Block}(\hat{a}))}\;\text{Hyp}}{\text{Big-Block}(\hat{a})}\;\text{Rips-Ande}}{\text{Big-Block}(b)}\;\text{Rips-Matching} \qquad \dfrac{\dfrac{}{(\text{Square-Block}(x) \Rightarrow \text{Green-Block}(x))}\;\text{Hyp} \qquad \dfrac{\dfrac{\dfrac{}{(\text{Big-Block}(\hat{a})\wedge \text{Square-Block}(\hat{a}))}\;\text{Hyp}}{\text{Square-Block}(\hat{a})}\;\text{Rips-Ande}}{\text{Square-Block}(\hat{a})}\;\text{Rips-Matching}}{\text{Green-Block}(b)}\;\text{Rips-Bckw-If-Elim}}{(\text{Big-Block}(b) \wedge \text{Green-Block}(b))}\;\text{Rips-Bckw-Andi}$$

Figure 7.10: PSYCOP proof of the example syllogism

Therefore, in order to show *Big-Block(b) AND Green-Block(b)* , it does not suffice to show *Big-Block(b)* and *Green-Block(b)* separately, but for the same individual represented by b.

The solution presented in this section, where a global constraint guides the instantiation of the temporary names, has similarities to the concept of a "variable condition" (like in [48]), which serves to record the quantifier ordering for quantifier-free representations of formulae. But the particular problem tackled here is a different one, namely that a temporary name in an open proof goal in quantifier-free form represents existential quantification. In classical natural deduction calculus like Gentzen's NK([20]), existential quantification is treated via *EXISTS Introduction* and *EXISTS Elimination*, and if two formulae share the same free variable, this variable name indeed identifies the same object. The semantic uniqueness of variables in classical natural deduction is not preserved in PSYCOP, since the same temporary name can be matched to different other temporary names. The mechanisms in PSYCOP and in this thesis thus both serve to propagate the identity of temporary names shared between formulae in quantifier-free form.

7.4.6 PSYCOP Example Problem

The most elaborate example for an application of a PSYCOP inference rule to a first-order logic problem in quantifier-free presentation is found in section "Soundness of Deduction Rules" on page 212 in [40]. The problem is stated in Figures 7.12 and 7.11 in classical logic notation and in quantifier-free notation.

$$\frac{\begin{array}{l}\forall x \forall y \forall z(\mathrm{In}(x,y) \wedge In(y,z) \Rightarrow In(x,z)) \\ \forall w \exists a(\mathrm{In}(w,a) \wedge \mathrm{In}(a,\mathrm{Africa}))\end{array}}{\forall u(\mathrm{In}(u,\mathrm{Africa}))}$$

Figure 7.11: Example in standard notation

$$\frac{\begin{array}{l}\text{IF In}(x,y)\text{AND In}(y,z)\text{THEN In}(x,z) \\ \text{In(w,}a_w\text{) AND In(}a_w\text{,Africa)}\end{array}}{\mathrm{In}(u, Africa)}$$

Figure 7.12: Example in quantifier-free notation

Note that in the case of the quantifier-free problem formulation in ΩMEGA , the "temporary name" a_w is formally expressed as a Skolem term of the form $a(w)$, i.e. $(a\ w)$ in LISP notation.

Proof line	Support lines	Formula	Inference Rule
A1.	A1	$\vdash [[In(x,y) \wedge In(y,z)] \Rightarrow In(x,z)]$	(Hyp)
A2.	A2	$\vdash [In(w, Tmp_{-2}(w)) \wedge In(Tmp_{-2}(w), Africa)]$	(Hyp)
L1.	A1, A2	$\vdash [In(u, Tmp_{-1}(u)) \wedge In(Tmp_{-1}(u), Africa)]$	(Rips-Matching A2)
C1.	A1, A2	$\vdash In(u, Africa)$	(Rips-Bckw-If-Elim A1,L1)

Figure 7.13: PSYCOP proof of the example as ΩMEGA derivation

Proof line	Support lines	Formula	Inference Rule
A1.	A1	$\vdash \forall X, Y, Z. [[In(X,Y) \wedge In(Y,Z)] \Rightarrow In(X,Z)]$	(Hyp)
L2.	A1	$\vdash \forall Y, Z. [[In(U_1,Y) \wedge In(Y,Z)] \Rightarrow In(U_1,Z)]$	($\forall E$ A1)
L6.	A1	$\vdash \forall Z. [[In(U_1,S_1) \wedge In(S_1,Z)] \Rightarrow In(U_1,Z)]$	($\forall E$ L2)
L7.	A1	$\vdash [[In(U_1,S_1) \wedge In(S_1,Africa)] \Rightarrow In(U_1,Africa)]$	($\forall E$ L6)
L4.	L4	$\vdash [In(U_1,S_1) \wedge In(S_1,Africa)]$	(Hyp)
L5.	A1, A2, L4	$\vdash In(U_1,Africa)$	($\Rightarrow E$ L4,L7)
A2.	A2	$\vdash \forall W. \exists S. [In(W,S) \wedge In(S,Africa)]$	(Hyp)
L3.	A2	$\vdash \exists S. [In(U_1,S) \wedge In(S,Africa)]$	($\forall E$ A2)
L1.	A1, A2	$\vdash In(U_1,Africa)$	($\exists E$ L3,L5)
C1.	A1, A2	$\vdash \forall Dc{-}_{315}. In(Dc{-}_{315}, Africa)$	($\forall I$ L1)

Figure 7.14: Gentzen natural deduction proof of the example

This example problem was proven in the ΩMEGA framework with PSYCOP's inference rule *Backward IF Elimination*, like in [40]. The proof is presented in Figure 7.13. An alternative proof in Gentzen's ND calculus was performed for comparison (Figure 7.14).

The proof in PSYCOP's calculus requires one rule application of *Backward IF Elimination* together with matching, as shown in Figure 7.13. As a part of the rule application to the proof goal $In(u, Africa)$, this goal is matched with $In(x,z)$, the consequent of the implication that expresses transitivity. The antecedent of the implication is then transformed as follows:

- x is substituted by u, and z is instantiated by $Africa$.
- The "argument reversal" procedure yields $In(u, Tmp_1) \wedge In(Tmp_1, Africa)$, where Tmp_1 is a new temporary name.
- The subscript reversal establishes the dependency of the temporary name Tmp_1 on the variable u. The new subgoal thus created constitutes line L1

in the proof.

In a second step (which is not part of the *IF Elimination*), the new subgoal $In(u, Tmp_1(u)) \wedge In(Tmp_1(u), Africa)$ is matched to the assertion $In(u, Tmp_2(w)) \wedge In(Tmp_2(w), Africa)$, and the proof is closed.

The alternative proof in Gentzen's ND calculus, presented in 7.14, is also centered around the *IF Elimination* rule, but it additionally requires four *FORALL Eliminations*, one *FORALL Introduction*, and one *EXISTS Elimination*. These steps are all represented implicitly in the matching process in the PSYCOP proof.

7.5 Discussion

There is general criticism on considering a rule system as the centerpiece of human deductive reasoning. Therefore, Rips has been attacked by opponents of rule-based theories. Nevertheless, the theory is one of the most comprehensive theories known today in the psychology of reasoning.

Rips's theory has soon attracted criticism by Philip Johnson-Laird, a proponent of the *Mental Model* theory. This has lead to a controversial exchange of ideas in several articles that Rips and Johnson-Laird were addressing to each other, namely [27], [41] and [28]. The main idea behind *Mental Models* is that human reasoners consider possible instantiations of the argument (possible "states of affairs", also referred to as "mental models"), which allows them to construct counterexamples if possible. Johnson-Laird argues in favor of his own theory that in the case of an invalid argument it is implausible that - as postulated by PSYCOP - subjects exhaustively search for a proof until they finally formulate the conclusion "doesn't follow". Johnson-Laird states that Rips's account of how subjects decide that an argument is invalid is "too weak". He also points out that Rips's theory also lacks explanations of how subjects systematically infer invalid conclusions.

On the other hand, Rips does not claim that his theory is an exhaustive theory on human reasoning. He considers his theory as "psychologically incomplete w.r.t. the wider field of (logical) operators" (cf. [40]). He does not see himself in the tradition of logicism, a movement that postulates that "everything is a matter of deductive reasoning" and ignores other modes of reasoning, such as probabilistic reasoning or analogy, for example. Thus, Rips escapes the criticism of Johnson-Laird against logicism.

Further criticism by Johnson-Laird does not only touch the *Deduction System Hypothesis* as such, but the choice of PSYCOP's inference rules. He doubts the "psychological plausibility" of the rule set chosen by Rips, in particular with respect to the quantifier-free representations of the rules for classical predicate logic. He points out that the formulation of the rules as procedures is rather lengthy and involves complicated sub-procedures, which contradicts the intuition that these rules shall be “intuitively valid”.

Lance Rips defends his theory against this attack. In [41], he argues that even though the formalization of the rules look complex, this need not imply that the rules are “hard for people to use, any more than a detailed description of the visual system implies that objects are hard for people to perceive” ([41], p. 413).

Even though it remains controversial which rules are exactly those that can be considered to be the “rules of the mind”, Rips presents a theory that - unlike other theories - gives a broad account on deductive reasoning of ordinary people. The parameters for the availabilities of the inference rules are in line with previous findings and support the idea of treating quantification implicitly.
Apart from providing the PSYCOP theory, Rips provides experimental techniques for evaluating the psychological plausibility of a theory on deductive reasoning, as well as numerous empirical results. Even though the *Deduction System Hypothesis* as such, as well as the concrete rules and heuristics in PSYCOP remain disputable, it becomes clear that proof calculi such as Gentzen’s natural deduction ([20]) can be considerably improved in modeling human reasoning abilities by psychological investigation and empirical studies.

Rips’s work also provides an example of a logical theory that contains redundant rules. That is, some of the rules do not add to the logical power of the system, only towards proofs of a smaller size since they provide shortcuts w.r.t. a more succinct rule system. For example, *Backward Disjunctive Modus Ponens* can be expressed by one *Backward IF Elimination* application and one *Backward OR Introduction*. By making a particular deduction argument a rule, Rips accounts for the relative ease of such a deductive argument.

The PSYCOP theory claims to be a general theory on logical reasoning, and Lance Rips illustrates this with several examples like a classification problem in the section “The Role of Deduction in Thought” in [40]. Even though the work does not explicitly mention the aspect of granularity of proof steps, a main concern of PSYCOP is to relate the performance of reasoners on a proof problem to the length of the “mental proof” it requires. And indeed, if the PSYCOP mechanism is supposed to reflect the basic functioning of logical thinking, then the PSYCOP proofs should account for all qualities of logical thinking, including granularity phenomena.

Lance Rips stresses that his study only concerns people untrained in formal logic, so his theory does not specifically account for phenomena in mathematical thinking. But also there, the question arises whether a succinct mathematical theory reflects the thinking process of a human mathematician. Another question is the role of specific factors in mathematics that influence the qualities attributed to proof steps, like mathematical notation, linguistic pitfalls, erroneous knowledge and pragmatic considerations. In his work on logic, Rips considers only one aspect that reaches beyond pure logics, namely pragmatic considerations. He assumes that if a reasoner does not find a proof for his argument, he will guess.

Apart from the principal question concerning the plausibility of PSYCOP w.r.t. human reasoning, the work in this thesis has investigated the challenges that arise when implementing the PSYCOP rules in a general framework for deduction systems. This has shown that the backward rules proposed in the PSYCOP theory are fairly complex procedures. Besides the high number of sub-steps required for the rules, the single steps themselves require subroutines that are highly nontrivial, like "argument reversal" and "subscript adjustment" procedures, the "matching" mechanism, and other procedures whose complexity is hidden in their formulation given by Rips as descriptions in English language. The LISP implementation for the ΩMEGA framework of "subscript adjustment" alone is a bigger and more complex LISP code fragment than any encoding of a backward inference rule specification as presented in [40]. Even if it can be assumed that similar computations are carried out by human reasoners with ease and without conscious awareness of the complexities, this is an obstacle for practical work with the PSYCOP theory. It remains yet an interesting question, which roles reductionism and parsimonious simplicity - as a criteria for theory choice - play in the investigation of deductive reasoning. Can we assume that the first principles that guide deductive reasoning are simple in nature?

Chapter 8

Analysis with the Granularity Framework

We now employ the granularity evaluation framework for an exemplary investigation of granularity phenomena. As the raw material for the investigation, we take the corpus collected in the Wizard-of-Oz Study introduced in Chapter 3. The dialog contributions that represent mathematical statements are considered as proof problems and formalized in the ΩMEGA framework. Then proof search is attempted for these examples, once according to the PSYCOP theory using the PSYCOP deduction rules for ΩMEGA provided in the context of this thesis, and also alternatively in the setting of Gentzen's natural deduction calculus [20]. The proofs obtained this way are then assigned a granularity measure using the granularity evaluation framework, and these parameters are compared to the granularity ratings of the wizards. As a first hypothesis, the number of inference steps needed to justify an inference drawn by the study subjects is assumed to represent a measure for granularity.

As a first step, we present how the proof problems from the experiment - together with the relevant assumptions from the study material - are modeled in the ΩMEGA framework.

The succeeding sections illustrate the generation of proofs for the mathematical statements and their systematic evaluation in the granularity framework. The results of the analysis suggest that the number of inference steps is not sufficient for a robust diagnosis, neither in the case of the PSYCOP theory nor in the case of Gentzen's natural deduction calculus [20]. Nevertheless, some interesting observations can be made.

8.1 Modeling of the Mathematical Domain

The set of mathematical theories in ΩMEGA was extended by the precise definitions in the study material handed out to the participants.

As described in Section 7.4.1, theorems and definitions are defined in the *POST* language in ΩMEGA. In principle it is not difficult to automatically convert the original mathematical formulae from the experiment to *POST* syntax. Even the more, because the instruction documents and the dialogs from the experiment are given in TeXmacs markup language which is similar to LaTeX markup. In the scope of this thesis, a simple parser was written which is able to convert formulae from the corpus into the *POST* language. It is based on the lexical analyzer *lex* and the parser generator *yacc*. Its main task is to convert the infix notation used for mathematical formulae in the corpus to the prefix notation in *POST*.

But for most dialog contributions in the experiment, parsing formulae is not sufficient to generate an accurate formalization of the mathematical statements. Most dialog contributions from the corpus require deep natural language analysis. The interesting research goal to develop parsing techniques for natural language with mathematical content represents a large part of the activity in the DIALOG project, but is beyond the scope of this thesis.

Figure 8.1 shows the definition of equality according to the extensionality principle, as it was presented to the subjects in the experiment. The corresponding definition in *POST* syntax is displayed below.

Since ΩMEGA is based on typed lambda calculus, every variable or constant has to be assigned a type when it is introduced. Note that the definition of equality only holds for relation types. Since ΩMEGA does not support polymorphic types, this particular definition of equality had to be used. Relations are assigned the type $i \to j \to o$, i.e. the type of a predicate on two individuals of types i and j.

The formalization of the relevant concepts in the study material already provide examples that show a gap between rigorous formalism and common mathematical practice. Figure 8.2 shows the formalization of the concept of relation composition. Note that only at first glance $\circ$ appears to be a binary operator. Implicitly, $\circ$ is defined on two relations on a particular set M, and the formal definition has to take this into account. The definition of the inverse relation (see Figure 8.3) is comparatively straightforward.

Das Extensionalitätsprinzip für Mengen besagt, daß zwei Mengen A, B genau dann gleich sind, wenn sie dieselben Elemente enthalten:
$A = B \Leftrightarrow \forall x(x \in A \Leftrightarrow x \in B)$.

"The extensionality principle for sets claims, that two sets A, B are equal if and only if they contain the same elements:
$A = B \Leftrightarrow \forall x(x \in A \Leftrightarrow x \in B)$."

```
(th~defdef relation=
     (in relation)
     (type-variables i j)
           (definition
              (lam (A (o j i))
                   (lam (B (o j i))
      (forall (lam (x i)
                        (forall (lam (y j)
   (equiv (A x y) (B x y)))))))))
(help "Equality on relations,
      defined by extensionality principle"))
```

Figure 8.1: Extensionality principle and its POST formalization

Das Relationenprodukt zweier binärer Relationen R, S in einer Menge M ist definiert durch
$R \circ S := \{(x, y) | \exists z (z \in M \wedge (x, z) \in R \wedge (z, y) \in S)\}$.

"The relational product of two binary relations R, S in a set M is defined by $R \circ S := \{(x, y) | \exists z (z \in M \wedge (x, z) \in R \wedge (z, y) \in S)\}$*."*

```
(th~defdef composition
(in relation)
(type-variables i)
(definition
 (lam (baseset (o i))
  (lam (rel1 (o i i))
    (lam (rel2 (o i i))
      (lam (x i)
(lam (y i)
          (exists
            (lam (z i)
              (and (and (baseset z)
        (rel1 x z))
     (rel2 z y))))))))))
(help "Definition of relation composition."))
```

Figure 8.2: Relation composition and its POST formalization

Die *inverse (konverse, reziproke, duale) Relation* von R ist die Relation $R^{-1} := \{(x,y)|(y,x) \in R\}$. "

"The inverse (converse, reciprocal, dual) relation of R is the relation $R^{-1} := \{(x,y)|(y,x) \in R\}$."

```
(th~defdef inverse
 (in relation)
 (type-variables i)
   (definition
    (lam (rel (o i i))
      (lam (x i)
        (lam (y i)
         (rel y x)))))
(help "Definition of the inverse relation;
(inverse R) is the relation S, such that Sxy iff Ryx."))
```

Figure 8.3: Relation inverse and its POST formalization

8.2 Modeling of the Proof Steps from the Corpus

Obtaining the formal representation of the proof steps from the corpus requires different kinds of processing. The natural language text needs to be semantically analyzed, with several aims:

- Identification of mathematical concepts.
- Determining the role of the mathematical statements for the proof task, e.g. discrimination between forward and backward steps, or between hypotheses and inference steps.
- Translation of the mathematical text to a suitable proof language.

This processing requires the deep, semantic analysis of natural language. The success in the disambiguation of different readings, in resolving underspecification and in constructing an appropriate representation of the problem will determine the effectiveness and accuracy of the later analysis of this "formalized" problem by a dedicated theorem prover.

In this little study, the interpretation of natural language text was done by hand by the author. This represents an idealized form of natural language analysis which is supposedly superior to mechanical analysis. But for this study, the assumption of an idealized natural language analysis reduces possible distortions of the granularity scores related to natural language. The study of natural language interpretation are outside the scope of this thesis.

On the other hand, many of the examples from the corpus that were analyzed contained no or hardly any words, so they required little or no natural language processing. For many of these extreme examples, it sufficed to parse the formula under investigation with the parser to $POST$ syntax and make it the conclusion of a new problem with the statements from the preceding utterances as assumptions. The occurrences of the letters x, y, z were interpreted as variables, and the occurrences of the letters R, S and T as binary relations on the set M, as introduced in the instruction material. All instances of $\circ$ were interpreted as relation composition on the underlying set M.

8.3 Case Studies: Gentzen's vs. Rips's Natural Deduction

The notion of 'natural deduction' already suggests that these calculi attempt to match the actual reasoning process by a human reasoner who is solving a problem. If a calculus represents the basic building blocks of thinking, then the "size" of a formal argumentation in terms of granularity should be reflected by the complexity of the steps in the calculus. Therefore, if a proof procedure plausibly reflects the grain size of actual mathematical practice, the measure for the size of proofs obtained that way should allow the discrimination between those proof steps that were "too coarse", "adequate" or "too detailed" in the eyes of the tutors.

In the course of this study, 20 proof steps from the corpus were converted into a formal problem in ΩMEGA, as explained above. These proof steps refer to the same exercise (exercise W, shown in Figure 3.5 in Chapter 3) to represent a homogeneous sample w.r.t. the mathematical task. All of these proof steps were taken from a pool of proof steps that have been identified as "correct" by the tutors.

For each of these proof steps, the formal problem statement was proven in ΩMEGA according to the PSYCOP theory and in Gentzen's natural deduction [20] calculus. The PSYCOP theory - based on natural deduction - specifies concretely how proof search proceeds, as introduced in Section 7.1.2. This represents a decision procedure that stops with either a proof or failure. Even though the rules in PSYCOP necessary to solve the problems were implemented in ΩMEGA, the heuristics were simulated by hand (which was not difficult, provided the small size of the proof problems). Gentzen's natural deduction as such does not specify heuristics for proof search. Therefore, different proofs are possible for the same problem. For this little analysis, only the shortest proof in terms of its number of proof steps was considered. These shortest proofs were again constructed by hand. For the proof steps from the corpus it was actually not difficult to see how to obtain the smallest proof that justifies a proof step in the corpus.

Neither PSYCOP nor Gentzen's natural deduction explain the role of definitions and defined concepts in mathematics. The simplest solution to systematically deal with definitions is to remove all definitions in the problem by replacing them with their definiens. This can be done with the help of an automatism in ΩMEGA, which also effectuates β-reduction after the replacements. The resulting representation would then be the basis for both the PSYCOP and the natural deduction proof. But the question is whether a definition expansion should be counted as a proof step, too, and therefore unnecessary definition expansions should be avoided in order to represent just the essence of a proof. Whenever possible, for each of the proofs in PSYCOP or Gentzen's natural deduction calculus a second version was constructed with only those definitions expanded that were needed to complete the proof. It is unclear if the number of definition expansions contributes more or less towards the complexity of a proof than the number of logical inferences, so both proof versions were provided for the analysis. Expanding all definitions always resulted in first-order logic problems that PSYCOP could solve, even though their original formulations contain higher-order constructs.

As a next step in the analysis, the number of proof steps was counted with the help of the granularity analysis framework introduced in Chapter 6.

Another question is whether those proof steps that do not contribute towards the goal should be counted towards the size of a proof. The PSYCOP heuristics introduces inference steps in the forward direction which might turn out to be useless. Therefore, one evaluation schema was counting only "necessary" proof steps, whereas another was counting all proof steps.

It should be noted that Rips's calculus was designed to reflect the abilities of people untrained in logics, and that parameter estimates were done with exactly this group of subjects. So even if the suitability of Rips's calculus for the domain of mathematics can be questioned, this should not generally invalidate his approach. It would rather motivate deeper investigations into finding a suitable representation for the granularity of mathematical proofs.

8.3.1 An Example Analysis

The dialog fragment given in Figure 8.4 provides a worthwhile example test case, for the following reasons:

- It contains a step that has been classified by the tutor as "too coarse-grained".
- The student hardly uses any language. Therefore granularity can be studied in the absence of linguistic factors.
- The sample shows the limits of PSYCOP's deductive power, not in the form of a toy example (like the paradox of implication in [40]), but in a real-world mathematical situation.

Utterances 1,5,6 and 7 are in principle "correct" mathematical statements. Nevertheless, from a strictly formal point of view Utterance "student 1" is incorrect, since x and y occur freely in the antecedent, whereas x and y are bound to the universal quantifiers in the consequent. So formally, x and y in the consequent denote objects different from x and y in the antecedent, which renders the statement unprovable in general. This error can be corrected easily, by moving the universal quantification of x and y to the very left. This was done for the analysis, but it cannot be expected that this would also be done by a mathematical dialog software.

The formal problem that was created from Utterance 1 is shown in Figure 8.5. This ΩMEGA problem definition contains definitions of the relations symbols R,S and T that appear in the exercise text in the instruction material. They are treated as constants with a relation type. For all of the mathematical statements student 0] $(R \circ S)^{-1} := \forall x \forall y \forall z[(x,z) \in R \wedge (z,y) \in S \rightarrow (y,x) \in (R \circ S)]$

tutor] Das ist richtig! *"That's correct!"*

teilw. korrekt	angemessen	relevant

student 1] $(y,x) \in (R \circ S) \rightarrow \forall x \forall y \exists z[(y,z) \in R \wedge (z,x) \in S]$

tutor] Das ist auch richtig! *"That's also correct!"*

korrekt	angemessen	relevant

student 2] $S^{-1} \circ R^{-1} := \forall x \forall y[(x,y) \in S \rightarrow (y,x) \in S] \wedge \forall z \forall w[(z,w) \in R \rightarrow (w,z) \in R]$

tutor] Das ist nicht richtig! Wenn $(x,y) \in S$, dann folgt nicht $(y,x) \in S$. *"That's not correct! If $(x,y) \in S$ then $(y,x) \in S$ does not follow."*

inkorrekt	angemessen	eingeschraenkt relevant

student 3] $S^{-1} := \{(x,y)|(y,x) \in R\}$?

tutor] Nein! $S^{-1} := \{(x,y)|(y,x) \in S\}$. Sie koennen Fragen auch in deutscher Sprache formulieren. *"No! $S^{-1} := \{(x,y)|(y,x) \in S\}$. You may formulate questions in German language, too."*

inkorrekt	angemessen	relevant

student 4] Danke. *"Thanks."*

tutor] Bitte! *"You're welcome!"*

k.A.	k.A.	k.A.

student 5] $(x,y) \in (S^{-1} \circ R^{-1}) \Leftrightarrow \exists z[(z,x) \in S \wedge (y,z) \in R]$

tutor] Das ist richtig! *"That's correct!"*

korrekt	zu grobschrittig	relevant

student 6] $\forall x \forall y[\exists z[(y,z) \in R \wedge (z,x) \in S] \rightarrow (y,x) \in (R \circ S)]$

tutor] Das ist auch richtig! *"That's also correct!"*

korrekt	angemessen	eingeschraenkt relevant

student 7] damit folgt $(x,y) \in (S^{-1} \circ R^{-1}) \rightarrow (y,x) \in (R \circ S)$ *"Therefore follows $(x,y) \in (S^{-1} \circ R^{-1}) \rightarrow (y,x) \in (R \circ S)$"*

tutor] Vielen Dank! Druecken Sie nun den Knopf fuer die naechste Aufgabe! *"Thank you! Now press the button for the next exercise!"*

korrekt	angemessen	relevant

Figure 8.4: Example dialog for exercise W. The tutor's ratings read as "korrekt": correct, "inkorrekt": incorrect, "angemessen": appropriate, "zu grobschrittig": too coarse-grained, "relevant": relevant, "eingeschraenkt relevant": restricted relevance, "k.A.": no comment.

```
(th~defproblem 0411-exercise-w-utterance-1
 (in relation)
 (type-constants i)
 (constants (R (o i i))
            (S (o i i))
            (T (o i i))
            (M (o i)))
  (conclusion step1
     (forall (lam (x i)
        (forall (lam (y i)
            (implies ((composition M R S) y x)
                     (exists
                        (lam (z i)
                            (and (R y z)
                                 (S z x)))))))))))
```

Figure 8.5: Example formalization from the example dialog

in the sample, these definitions were repeated in the problem statements. The formula labeled as a conclusion and named "step1" contains the formula of the utterance, transformed to prefix *POST* notation by the parser. Furthermore, it has been annotated with types, as required by ΩMEGA's type system. For each proof step, the previous correct proof steps were given as assertions. Since the particular example of Utterance 1 represents the first "correct" step in the proof, the problem statement contains no assertion.

It is also possible to interpret "→" in Utterance 1 in such a way that the antecedent becomes an assertion and the consequent the conclusion of the problem. This affects the number of proof steps needed to solve the problem. The assertion-conclusion relation in a proof problem implicitly represents an implication. In contrast to that, in the case of the formalization shown in 8.5 the implication has to be explicitly eliminated in the course of the proof. It can be argued for both of these interpretations of "→", as long as they are applied consistently. But it should be pointed out that this is an example for a situation where a certain degree of freedom in the interpretation and modeling of the proofs was encountered, which can potentially distort the granularity analysis.

In a next step, the occurrence of the predicate "composition" is replaced by its defining term. Note that the definition of "composition" on sets and relations is a higher-order predicate. The pure textual replacement with the body of the definition still produces a higher-order term. Only after β-reduction by the ΩMEGA-definition expansion mechanism, the antecedent of the implication becomes the first-order term

(exists (lam (z i) (and (and (M z) (and (R y z) (S z x)))))) ,

which is the *POST* representation for the formula

$$\exists z_i(z_i \in M \wedge (y, z_i) \in R \wedge (z_i, x) \in S).$$

Then, the PSYCOP deduction mechanism is applied to the problem. In this case, the problem can be solved via *Backward IF Elimination*, *AND Elimination* and *Backward AND Introduction*. The number of *AND Elimination* and *Backward AND Introduction* steps depends on the syntactic representation of the definition of composition, were the conjunction $(z \in M \wedge (y, z) \in R \wedge (z, x) \in S)$ can be either represented as $((z \in M \wedge (y, z) \in R) \wedge (z, x) \in S)$ or $(z \in M \wedge ((y, z) \in R \wedge (z, x) \in S))$. Whether $\wedge$ is left or right-associative is not defined in the instruction material. For the analysis, the parser converting the raw formulae to *POST* representation consistently treated $\wedge$ as left-associative.

Utterance	Proof Step Count		Tutor's granularity rating
	PSYCOP	Gentzen's ND	
1	7	9	A
5	5	3	C
6	2	3	A
7	10	9	A

Figure 8.6: Proof step ratings for the example dialog. Granularity ratings read as "C": too coarse, "A": appropriate, "D": too detailed.

The worrying issue is that here an aspect of the syntactic representation of the problem - which is not present in the original formula created by the student - affects the proofs in both natural deduction systems, and yields different hypothetical granularity results.

The results of the granularity analysis for this particular dialog are presented in Figure 8.6. It is part of the complete data set given in Appendix E and discussed in Section 8.3.4. In this particular case, the tutor rated proof step 5 as "too coarse". But in terms of the size of their justifications, the Utterances 1 and 7 require the largest proofs in PSYCOP and Gentzen's natural deduction calculus. So in this case counting the proof steps would be misleading and fail to identify Utterance 5 as "too coarse-grained".

8.3.2 Incompleteness of PSYCOP and further Observations

Even though a PSYCOP proof was found for all 20 problems in the sample, the incompleteness of PSYCOP had an effect on these proofs. Some subproofs of the proofs in Gentzen's natural deduction calculus for the sample problems could not be proven in PSYCOP. For some proof problems it was possible to obtain a proof in Gentzen's natural deduction calculus with still most definitions remaining unexpanded, but not in PSYCOP. Nevertheless, the PSYCOP mechanism always produced one proof. This is because the problem specifications created from the utterances often contained logical redundancies in their assertions which enlarged the proof options. The set of possible proofs was further enlarged when all the definitions were removed.

One illustrating example is Utterance 7 in Figure 8.4. At this point in the dialog, the student refers to the preceding steps in the proof by saying "therefore follows...". For the analysis, all the correct mathematical statements before Utterance 7 were provided as assertions for the problem representation.

In the PSYCOP proof, the mathematical statement $(x, y) \in (S^{-1} \circ R^{-1}) \Leftrightarrow \exists z[(z, x) \in S \wedge (y, z) \in R]$ which stems from Utterance 5 in Figure 8.4 is replaced by two implications. Then starting from the proof goal $(x, y) \in (S^{-1} \circ R^{-1}) \rightarrow (y, x) \in (R \circ S)$ the new proof goal $(y, x) \in (R \circ S)$ is created under the hypothesis $(x', y') \in (S^{-1} \circ R^{-1})$. In the process of *Backward IF Introduction*, new variables x' and y' are introduced for the variables x and y. Therefore, the new hypothesis $(x', y') \in (S^{-1} \circ R^{-1})$ is not identical to the left-hand side of the implication $(x, y) \in (S^{-1} \circ R^{-1}) \Rightarrow \exists z[(z, x) \in S \wedge (y, z) \in R]$ from Utterance 5. Though in spite of logical equivalence between $(x, y) \in (S^{-1} \circ R^{-1})$ and $(x', y') \in (S^{-1} \circ R^{-1})$, it is not possible to apply *Forward IF Elimination* to the implication $(x, y) \in (S^{-1} \circ R^{-1}) \Rightarrow \exists z[(z, x) \in S \wedge (y, z) \in R]$ and $(x', y') \in (S^{-1} \circ R^{-1})$. The reason is that the *Forward IF Elimination* rule is only applicable if the hypothesis and the antecedent of the implication are identical (i.e. matching between the hypothesis and the antecedent is not allowed).

This problem is a more complex version of an example for the incompleteness of PSYCOP given by Lance Rips in [40] on pages 193 and 194. There he considers the assertions "*IF Aardvark*(x) *THEN Mammal*(x)" and "*Aardvark*$($*Ginger*$)$", from which the conclusion "*Mammal*$($*Ginger*$)$" logically follows in classical logic with *modus ponens*. This step requires instantiating the variable x in the implication with the constant *Ginger*. The same could be achieved in PSYCOP if the *Forward IF Elimination* rule is allowed to match the antecedent of the implication to another assertion. But Rips explains that in this case, the self-constraining nature of *IF Elimination* is no longer given. He gives the following example: Assume a recursive definition of the predicate "*integer*", where "*integer*(0)" holds and "*integer*(x)" implies that there exists a successor of x that is again an integer. This implication leads to an infinite number of conclusions, if it can be applied to any "*integer*" with *IF Elimination*.

This is indeed the idea behind recursive definitions of infinite sets like the natural numbers, but without constraining the PSYCOP theory the inference mechanism becomes undecidable. Lance Rips's way of constraining the theory consists in disallowing matching for *Forward IF Elimination*, i.e. either antecedent and premise are syntactically equal - then *Forward IF Elimination* can be applied - or the rule application fails.

$$\frac{\begin{array}{l}\text{man(Socrates)}\\ \forall x\,(\text{man}(x) \Rightarrow \text{mortal}(x))\end{array}}{\text{mortal(Socrates)}}$$

Figure 8.7: Classical syllogism unprovable in PSYCOP

As a consequence, several classical examples of deductive arguments can not be proven in PSYCOP. As a popular instance of such a deductive argument, take the famous syllogism displayed in Figure 8.7.

In the case of Utterance 7 in Figure 8.4 a PSYCOP proof was finally found after the definitions in the formulae $(x, y) \in (S^{-1} \circ R^{-1})$ and $\exists z[(z, x) \in S \wedge (y, z) \in R]$ were replaced. But this proof does not include the assertion given in Utterance 5 in Figure 8.4. Therefore, it becomes doubtful that in the case of Utterance 7 the PSYCOP proof reflects the proof idea by the student, since she explicitly referred to her previous inferences.

However, the mathematical statement of Utterance 7 can be proven in Gentzen's natural deduction calculus using the assertions from the previous utterances. Nevertheless, nine steps are required for the proof, which is clearly higher than the average for the sample. This does not fit the tutor's verdict, who categorized this mathematical statement as "appropriate" with respect to granularity.

It should be noted that the proof in Gentzen's natural deduction calculus proceeds by deriving $\exists z[(z, x) \in S \wedge (y, z) \in R]$ from $(x, y) \in (S^{-1} \circ R^{-1})$ justified by Utterance 5, and then $(y, x) \in (R \circ S)$ with Utterance 6. However, the proof requires the commutativity of $\wedge$ to use the the term $\exists z[(z, x) \in S \wedge (y, z) \in R]$ from Utterance 5 together with the antecedent of Utterance 6, the term $\exists z[(y, z) \in R \wedge (z, x) \in S]$. In Gentzen's natural deduction calculus, this can only be achieved by decomposing the terms using *EXISTS Elimination* and *EXISTS Introduction* before proceeding with the application of commutativity.

This points to the fact that the natural deduction calculi considered here are only to be applied to the main connectives of formulae, not inside of them. The effect of this is less strong for the PSYCOP calculus. In explicitly quantified formulae, the number of quantifiers contributes to their depth, so their absence in quantifier-free formula makes the formulae more shallow. However, this a fundamental difference of both of these natural deduction calculi w.r.t. rewriting systems, for example. Observing the students while they were constructing the formulae provides hints that they were indeed manipulating sub-terms of formulae locally, more akin to deep inference techniques.

Another observation concerns the way the students reasoned about equivalence and equality, and how this was represented in the proofs constructed during the analysis. One mathematical statement from the corpus states that

$$(x, y) \in (R \circ S)^{-1} \Leftrightarrow (y, x) \in (R \circ S).$$

The corresponding justification in Gentzen's natural deduction calculus required three steps. The application of the definition of the inverse relation (which is not counted among the natural deduction steps) yields the term

$$(y, x) \in (R \circ S) \Leftrightarrow (y, x) \in (R \circ S).$$

At this point, the statement is not yet proven. The equivalence, which is treated as a defined concept, is expanded into two implications - both are trivially satisfied, but require *IF Introduction.* The PSYCOP proof is obtained in a similar way, but further requires two applications of matching. However, a mathematically trained person might not consider these steps as necessary, given that $A \Leftrightarrow A$ holds for any proposition A. However, providing the assertion $A \Leftrightarrow A$ and instantiating the variable A with a term is not possible in PSYCOP - matching is restricted to variables, temporary names and constants, which excludes composed terms.

Similarly, many of the formulae containing equality and equivalence can be tackled using the properties of equivalence relations. However, the natural deduction calculi considered in this thesis do not offer the possibility to apply such an asserted property directly (i.e. in one single inference step).

8.3.3 Discrimination of Granularity Levels

The question is whether the analysis of the proofs constructed for the mathematical statements in the corpus allows a classification into the three granularity classes "too detailed", "appropriate" and "too coarse-grained".

The method of statistical analysis that is concerned with the assignment of cases or events to different classes is called discriminant analysis. The score on a dependent variable in an experiment is related to a set of "explaining" independent variables. In this particular case of the granularity analysis, the dependent variable of the experiment is the classification by the tutor. There, the three classes of granularity “too detailed”, “appropriate” and “too coarse-grained” represent an ordinal scale, i.e. there is an ordering among the different levels of granularity. For this analysis, the size of the justifications for the mathematical statements w.r.t. to granularity as determined by the granularity analysis is considered as the “explaining” independent variable.

In typical applications of discriminant analysis, several discrimination functions are determined, always one less than the number of classes (cf. [13]). The discrimination functions are constructed using a learning set from the sample. Their effectiveness for classification is then tested by applying them to another “test set”. The quality of the discrimination is determined statistically from the mismatch between the a-priori classification and the classification with the discrimination functions.

As a first sample, 20 utterances from the experimental corpus (as presented in Chapter 3) have been analyzed w.r.t. the length of proofs that justify them. The question is whether the lengths of the proofs are indicators for their granularity as judged by the tutors. A second question is whether proofs according to the PSYCOP theory are better suited as indicators than proofs in Gentzen’s natural deduction calculus [20].

8.3.4 Results

General Observations

Proving the sample of twenty mathematical statements required on average 5.80 proof steps in a PSYCOP proof if all definitions had been removed a priori, and on average 6.00 proof steps if as many definitions as possible remained in the problem statement. In this score, the application of matching was counted as one rule application, as long as it was not part of another inference rule application. If matching is not considered as a proof step, the scores are 3.65 and 3.80, respectively. The average proof step count for Gentzen’s natural deduction calculus on the sample problems is 5.30 and 5.45, respectively.

Granularity Levels

In the entire corpus, only 20 statements were classified as “too detailed” and 51 as “too coarse-grained”, which is only a small portion of all statements. The vast majority was rated as “appropriate”. The sample of 20 statements analyzed in more detail reflects this imbalance, since only five of the twenty steps are “inappropriate”. The detailed scores for the 20 proof steps are given in Appendix E. The sample does not contain statements which do not represent any mathematical content or which were classified as incorrect. One example is the following dialog contribution, which was rated as “too coarse-grained”: “May I try to put mathematical relationships into words?”.

A graphical representation of the number of proof steps necessary to justify a statement from the corpus with PSYCOP and Gentzen’s natural deduction calculus is given in Figures 8.8 and 8.9. These figures depict the frequency distribution of the proof statements in the sample of twenty with respect to the size of their justification (in terms of the number of proof steps), which is marked on the x-axis. The categories assigned by the tutors to these statements are indicated by the shading. For example, Figure 8.8 shows that there were two statements from the category of “too detailed” steps and four statements classified as “appropriate” which were justified in PSYCOP by one step.

If the proof strategy of only expanding as many definitions as necessary is considered, the result in terms of the number of needed proof steps is only slightly different, as can be seen in the data set presented in Appendix E. The observer recognizes the central tendency of proof steps classified as “too detailed” to require smaller justifications. For PSYCOP, on average one proof step was needed to justify a “too detailed” mathematical statement, 5.26 steps were needed to justify an “appropriate” statement, and 11.67 steps were needed for a “too-coarse grained” statement in the PSYCOP theory (when all definitions were removed from the problems, and matching steps were counted as proof steps).

But looking at the figures also shows that the number of justifying proof steps is distributed very unevenly. For the data in Figure 8.8, there is a standard deviation of 6.8 proof steps for the “too coarse-grained” group, and a standard deviation of 4.88 for the “appropriate group”. However, the standard deviation for the “too detailed” group is 0. But with only 2 mathematical statements in the “too detailed” group, the standard deviation is little meaningful.

The high standard deviation for the number of proof steps that justify a statement in a given category - about nearly as high as the mean value - shows that the distribution is very different from a normal distribution, for example.

In both cases of PSYCOP and Gentzen’s natural deduction, the attempt of constructing a discrimination function for the sample on the basis of counting proof steps is thus bound to failure. In the case of PSYCOP, how can the statements from the “too detailed” group be distinguished from those statements in the “appropriate” group that require only one justification step? Any discrimination function would have to make wrong assignments already for the training set, not to speak about a possible test set. In general, discrimination between different groups makes only sense if they have little variation within the group and little overlap with other groups.

It appears that the data obtained from the granularity analysis with the simple mechanism of counting proof steps in the PSYCOP theory or Gentzen’s natural

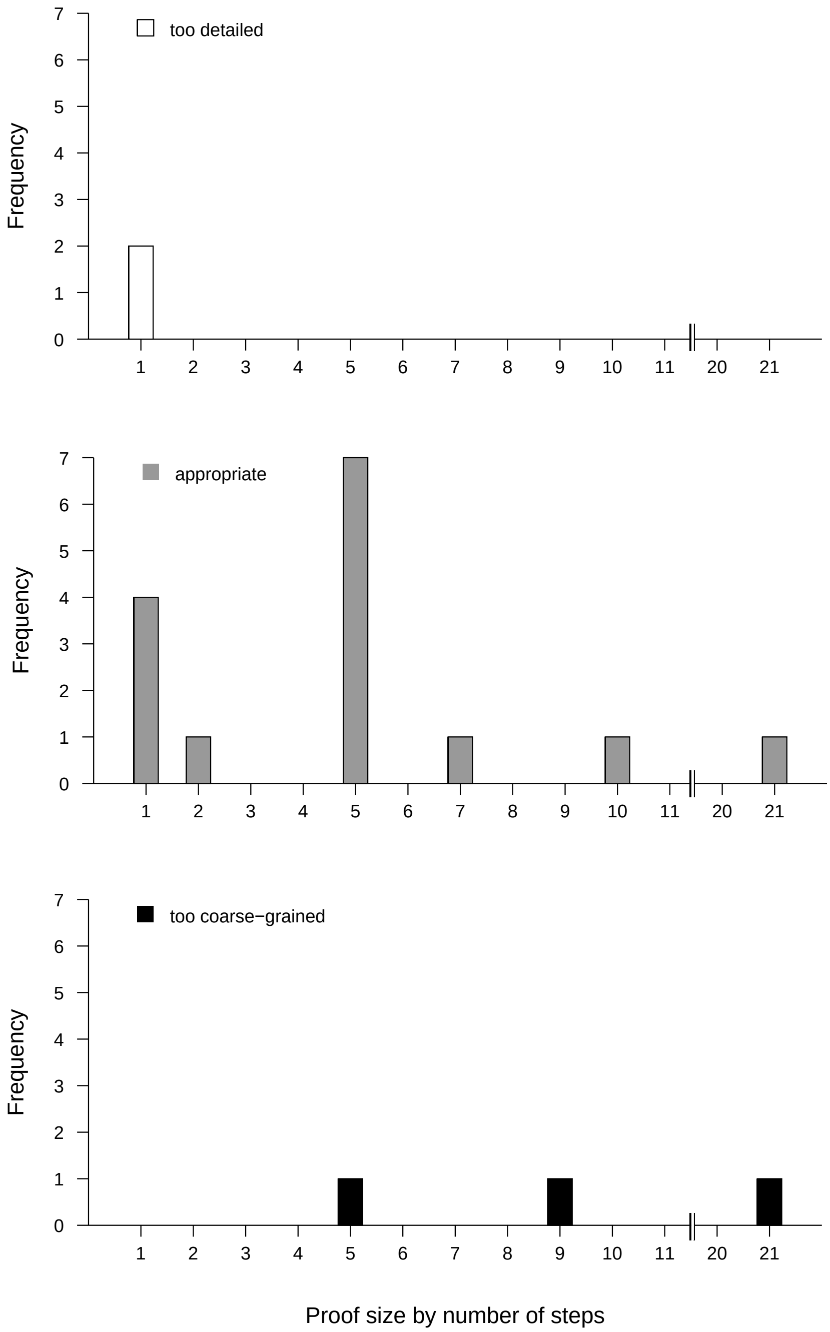

Figure 8.8: Frequency distribution of proof sizes - PSYCOP. Tutor's ratings are indicated by shading.

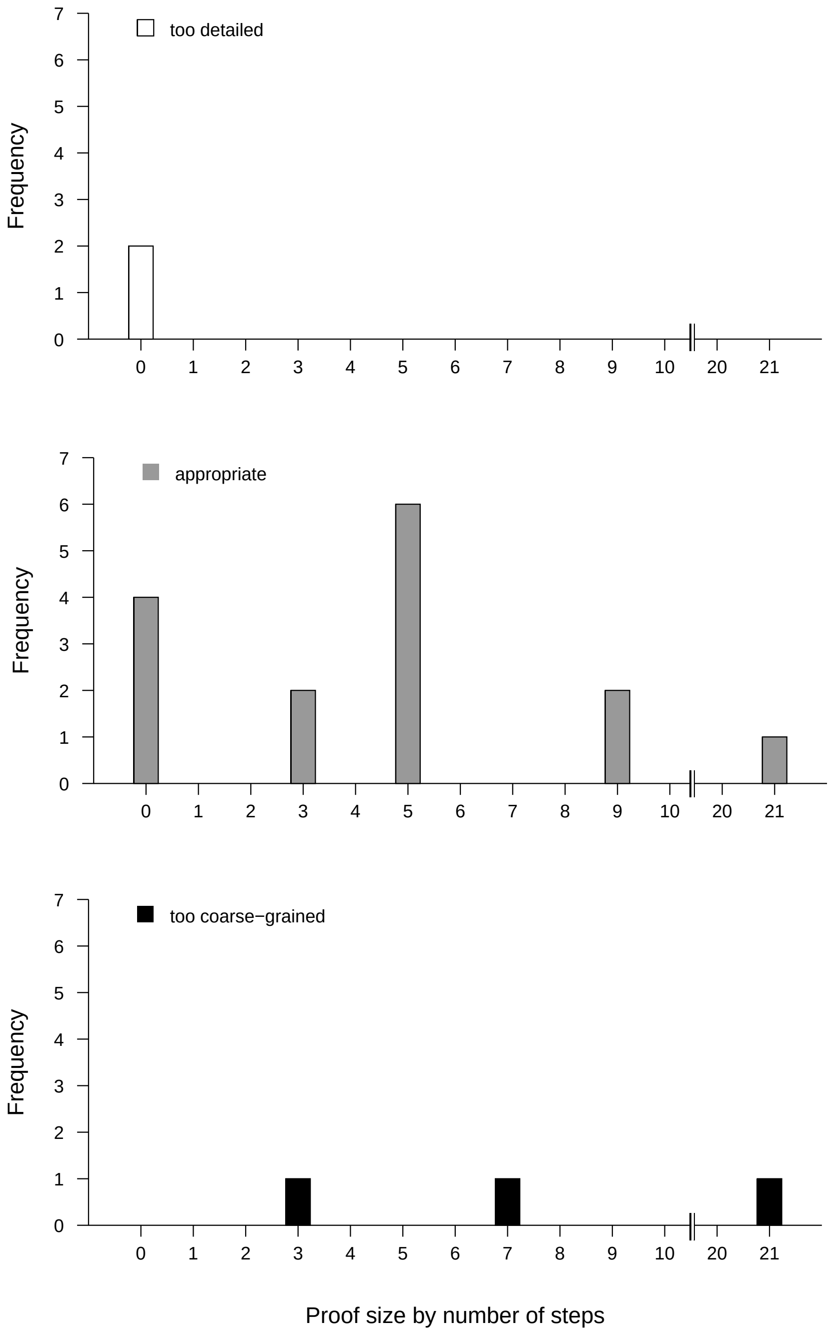

Figure 8.9: Frequency distribution of proof sizes - Gentzen's ND. Tutor's ratings are indicated by shading.

deduction calculus is too fuzzy to make a clear distinction between the different granularity classes, even though the central tendencies are visible. A deeper discrimination analysis might be suitable to provide further results. As a side remark, the simple strategy of classifying any statement from the corpus as "appropriate" will classify most of the statements in the corpus correctly. This is based on the simple observation that "appropriate" statements were far more common than "not appropriate" statements. In the case of this analysis, a classification strategy can only be considered useful if it performs better than such a simple strategy based on relative frequencies of the different classes. By looking at the data of the 20 proof statements in this preliminary analysis, this seems to be unattainable for the simple evaluation hypothesis of counting proof steps in the two natural deduction calculi.

8.4 Discussion

The analysis presented here is based on the very simple hypothesis that the granularity level of a mathematical statement in the corpus of mathematical dialogs as identified by the human experts can be determined from the number of proof steps needed to reconstruct such a step. This was tested for two different ways to represent the reconstructed proof steps, namely the PSYCOP framework and Gentzen's natural deduction calculus. Very quickly it became apparent that the naive hypothesis employed here does not allow a clear classification of the three granularity levels in the corpus. The statistical data shows a tendency of the three groups of proof steps to require justifications in PSYCOP and natural deduction with different proof sizes, but this tendency is not clear enough to provide a rigid distinction.

Several assumptions have been made to simplify the analysis but which may restrict the external validity of this study. First of all, the interpretation of the natural language and dialog aspects has been assumed to have an ideal performance, which can not be expected for an automated analysis. Secondly, in the case of the natural deduction calculus only the smallest reconstruction of the arguments w.r.t. to the number of proof steps was considered. This assumes the certainty that there will be no smaller proof, which can only be guaranteed by exhaustive search. Heuristics do help to find reasonably small proofs without searching excessively, but they provide no guarantee. For the small mathematical examples presented here there were little proof alternatives, and the analysis was not bound to particular time or resource limitations. But if such analysis is to be integrated in a working mathematical assistant system, computation time is an issue. This was demonstrated in the evaluation of the hypothetical "mathematical tutoring system" in the experiment presented in Chapter 3, where the waiting time for system responses was a main criticism from the participants.

It is also an open question whether "small proofs" are generally more reasonable to consider as explanations than other possible proof variants. The definition of the "size" of a proof already allows some freedom. For example, the size of a proof can also be measured in terms of a supposed "difficulty", rather than the number of proof steps.

The corpus would have allowed to extend the analysis to a larger sample of proof steps, but the first twenty examples already showed that the approach is not suitable for cleanly discriminating the different granularity classes. A thorough discriminant analysis might provide more indications, even though it appears unlikely that the general picture will change.

When looking at each particular mathematical statement in the sample, their justifications in PSYCOP or Gentzen's natural deduction calculus appear somewhat orthogonal to the original formulations. Several reasons can be given:

- Neither PSYCOP nor Gentzen's natural deduction explain the role of definitions and defined concepts in mathematics. However, definitions were mentioned frequently in the explanations given by the participants.
- Neither PSYCOP nor Gentzen's natural deduction support rewriting (like equality substitutions in formulae).
- At some points in the proofs, inferences in higher-order logic might provide more natural explanations.
- PSYCOP is incomplete - even though for the small sample, some proof has always been found in PSYCOP, the suitability of PSYCOP for mathematics is still in question. It appears unreasonable to assume that the logical incompleteness of untrained human reasoners also applies to mathematicians.
- Some assumptions used by the subjects are neither in the study material, nor made explicit - but supposedly belong to the common ground that the subject and the wizard share (like, for example, the transitivity and symmetry of $=$). For the design of an accurate on-line analysis mechanism for granularity, these assumptions will have to be anticipated by the exercise designer.

Chapter 9

Conclusion and Further Directions

The investigation into the notion of granularity in mathematical proofs as part of this thesis, as well as the experiment in the Wizard-of-Oz paradigm and its analysis with the proposed analysis framework provide the basis for potentially deeper investigations. The aim of this chapter is to sum up the major contributions and findings from this thesis. Furthermore, an outlook is given on the possible further development of mechanisms that analyze granularity in mathematical proofs. The chapter closes with the general discussion of the approach taken in this thesis.

9.1 Results and Achievements

The study of the characteristics of different grain sizes in mathematical proofs in this thesis has led to the design of the Wizard-of-Oz experiment. The experiment represents an exploratory study centered around the two main aspects of granularity and the analysis of natural language in mathematical dialog. The experiment provides a large corpus of 37 natural language dialogs in a non-trivial mathematical domain. Along with the dialogs come a variety of audio and video recordings, as well as comprehensive pre- and post-test questionnaires. Furthermore, the mathematical statements by the students were annotated with ratings from four expert tutors in mathematics. The corpus allows the study of a variety of phenomena related to the tutorial dialog on mathematics. It demonstrates that the issue of inappropriate grain sizes of mathematical statements were indeed subject of criticism by the tutors. The corpus also allows to analyze the mathematical style chosen by the subjects in contrast to the style advocated by the tutors. Furthermore, the experiment demonstrates the effect of two different sets of mathematical instruction material on the dialog, w.r.t to the use of natural language and the mathematical approach by the subject.

Apart from the design, carrying-out and a first analysis of the experiment, this thesis presents a framework to analyze mathematical proofs by annotating proof steps and proof statements with parameters which can be flexibly combined. This framework was used to test the hypothesis that the size of a mathematical argument in terms of the number of proof steps can serve as an indicator for the argumentative size attributed by a human tutor. This hypothesis was tested for the PSYCOP theory [40] and Gentzen's natural deduction calculus [20]. The thesis also includes a thorough study of the PSYCOP theory and a representation and implementation of most of the deduction rules in the ΩMEGA system, so that it can be studied along with Gentzen's natural deduction calculus in the same framework.

A first analysis of 20 proof arguments from the corpus shows that a clear discrimination between the three considered granularity levels with the described approach appears to be out of reach, both for proofs according to the PSYCOP theory and proofs in Gentzen's natural deduction calculus. In the analysis, discrepancies between proof steps in the corpus and their representation in the natural deduction calculi become apparent. This puts the suitability of proofs in natural deduction calculi as a representation for human-made proofs in question. The study also points to a number of features missing in the PSYCOP theory to make it suitable for mathematics. The PSYCOP theory is intended to reflect the reasoning abilities of people untrained in formal logics, and clearly requires adjustments to account for reasoning by mathematically trained people.

On the other hand, the collected corpus shows that there are dialog contributions where the tutor's verdict concerning granularity is not sufficiently explained in terms of purely logical arguments alone, but rather with regard to other factors, such as natural language as a means to explain proofs.

9.2 Further Directions

As to date, there is no formal calculus that is both shown to be psychologically plausible and yet powerful enough to represent non-trivial mathematics. The findings from this thesis further underline this statement.

There are candidate proof techniques which may potentially have an advantage over Gentzen's natural deduction calculus [20] and the PSYCOP theory. One candidate is the technique of *proof planning*, associated with the name of Alan Bundy [14]. Another candidate is the *assertion level proof* technique proposed by Xiaorong Huang [26]. A third candidate is the CoRe calculus by Serge Autexier [4].

Thus it is also an open question, whether it will be possible to formulate a psychologically adequate calculus as a concise and elegant system, or if it is required to incorporate many different, problem-specific techniques into it. What makes the problem particularly hard is that the hypothetical psychological plausibility can not be justified rigorously, but will require empirical results, since the inner functioning of the human brain is hidden from direct investigation. What makes the problem even harder is the fact that the skills for doing mathematics are acquired in a long learning process that ranges from early lessons in school to highly specialized courses at university. Whereas Lance Rips claims, among others, that logical reasoning on a general level is a central and innate capability of human beings, non-trivial mathematical reasoning appears to be a very particular skill that might be characterized differently among individuals. Different modes of reasoning (e.g. diagrammatic and deductive) and the notion of "intuition" in mathematics point to the possibility that different individuals might use different approaches to mathematical problems, according to their personal preferences. Unlike many formal deduction systems, such techniques may exhibit redundancies or inaccuracies. Therefore, a "psychologically plausible" calculus will not be more than an approximation to general mathematical practice.

Another limitation of the approach to granularity presented in this thesis is that only one possible proof alternative for a mathematical statement was considered as its explanation. This supposes that an automated theorem prover should be capable of finding the "most plausible" justification of a proof argument in one shot. But on the other hand, it is useful to consider many different proof alternatives for one statement - it is not always clear a priori how to find the "most plausible" proof, and the knowledge about the number and qualities of possible proof alternatives can potentially influence the judgment about an argument as well. Therefore, a more advanced technique for the analysis of mathematical statements is the enumeration and analysis of a variety of different proofs for the same problem. This was implicitly done in Chapter 8 when probing how many definitions in the a statement had to be replaced to find the "shortest" proof. The explicit analysis of different proof alternatives has not been attempted so far.

It should also be pointed out that the analysis of linguistic aspects, which have potentially contributed to the granularity ratings of the tutors, are not in the scope of this first approach to granularity presented here. However, the consideration of linguistic aspects, in particular with respect to explicitness and underspecification phenomena, seems to be required for a more comprehensive account of the tutor's verdicts.

The analysis of the proof steps during the experiment was done by the tutors and later compared to the outcome of the mechanical analysis with the framework. This contributes to the development of mechanical analysis procedures, but it still needs to be shown whether such procedures will be reasonably fast and robust when used as a part of a dialog system. The investigation of the aspect of granularity in mathematical proofs was motivated by the aim to build a granularity diagnosis module that will support the dialog management or a tutoring component in a larger system with specific information. Using the granularity analysis framework together with a suitable automated proof procedure in a working system will allow to shift away from Wizard-of-Oz experiments to practical experiments and a direct evaluation. Such a component can be used in a tutoring system for mathematics, but potentially also to provide mathematical domain knowledge for proof verbalization at different levels of granularity.

9.3 Where are we now?

The dispute between Lance Rips and Philip Johnson-Laird outlined in Section 7.5 is in the tradition of an old and yet unresolved quest in cognitive psychology, namely to characterize those processes in the human mind that produce logical reasoning. For the sake of this thesis, two rule-based calculi (namely PSYCOP [40] and Gentzen's natural deduction [40]) were chosen to represent deductions - mainly because the formal properties of rule-based systems are well investigated, and because their implementation is more straight-forward than other approaches.

But it is still open, whether a general-purpose rule system provides a more adequate representation of human reasoning than a collection of several domain-specific mechanisms. Rips's claim, that it is one rule system that represents logical abilities on a variety of tasks, is emphasized by examples. There, his system PSYCOP is used in different domains, like problem solving in artificial intelligence, logical puzzles, and a classification task (cf. Chapter 8 in [40]). But he admits that the PSYCOP theory might be "psychologically incomplete w.r.t. the wider field of (logical) operators" (cf. [40]). The emergence of a variety of approaches to logical reasoning, like the diagrammatic reasoning approach with "Spider Diagrams" (cf. [18]), or the model-based approach taken by the "Mental Models" theory, suggests that it will be unlikely that either one of them alone will provide a satisfactory account of logical reasoning in humans. The findings from this thesis can be interpreted along the same lines. It might have seemed more natural to express many steps from the corpus in a term rewriting system, but for many others natural deduction was the right choice. The PSYCOP theory contains redundant rules that are included only because it seems that these rules represent "natural" reasoning steps.

But the approach of including more and more - potentially redundant - rules into a calculus makes it computationally less tractable - unless new restrictions are introduced. A proof system with a high degree of redundancy will also allow a larger set of different proofs for the same problem, where some proofs are potentially just reformulations of other, equivalent proofs. The enumeration of proofs will be a challenge w.r.t. runtime and space requirements. It is unlikely that there will be an easy way out. On the other hand, a human mathematician is very skillful in maintaining a large mental knowledge base on mathematical facts and techniques without getting lost in the complexity. This gap is still to be bridged.

One of the central ideas for using the dynamic analysis of proofs in dialog systems - as opposed to using pre-formulated mathematical knowledge supplied by mathematical experts - is its flexibility. Flexibility is one of the requirements for a domain reasoning module in an intelligent tutoring system. But in order to make the study presented in Chapter 3 a controlled experiment, only one particular domain of mathematics was considered. But it is unrealistic to expect that the users of a mathematical dialog system will all have the same common background, as enforced in the experiment, and that it will only be used for a particular domain. The advantage of a dynamic proof evaluation over a system that uses only "golden solutions" still has to be shown.

Another aspect, which is also not touched by this thesis, is the pedagogical value of different approaches to mathematics. As an example from the thesis, the tutors suggested the students to use an approach based on the extensionality principle for the first exercise in the experiment, even though other styles were also applied successfully. Why did the specialists favor one style over the other in particular situations? This is of high interest when teaching mathematical proofs, but also for the design of an intelligent tutoring system. One researcher who is devoted to the cognitive aspects of proof is David Tall. In one study [46], he presented different proofs of the irrationality of $\sqrt{2}$ to 33 beginner students in mathematics and asked them which ones they understand. He repeated this procedure with more general proofs on the irrationality of arbitrary square roots. Tall concludes that the majority of students in his study prefers proofs with a supposedly high explanatory power over elegant proofs and general proofs. He describes that when asked about the reasons for their responses, the students referred to "unfamiliarity causing lack of understanding and confusion, whilst familiarity was linked with understanding - a far cry from the strict application of logic being the criterion for a good proof." (cf [46], p. 4). Since there is no known algorithm to generally analyze the "explanatory power" of a proof, this kind of analysis seems to be out of reach. But beyond that, this work raises the general question which properties of proofs are relevant for the learning of mathematics that are not captured in the structure of the formal arguments ("the strict application of logic", with Tall's words).

This thesis provides a first step into an area of research in the point of intersection between formal mathematics and cognitive science that calls for much further exploration.

Bibliography

[1] SFB 378. Collaborative research centre 378: Resource-adaptive cognitive processes. http://www.coli.uni-saarland.de/projects/sfb378/.

[2] John R. Anderson, C. Franklin Boyle, Robert Farrell, and Brian J. Reiser. Cognitive principles in the design of computer tutors. In P. Morris, editor, *Modelling Cognition*. 1987.

[3] Ken Appel and Wolfgang Haken. Every planar map is four colorable. *Illinois Journal of Mathematics*, 21:429–567, 1977.

[4] Serge Autexier. *Hierarchical Contextual Reasoning*. PhD thesis, Computer Science Department, Saarland University, Saarbrücken, Germany, december 2003.

[5] Serge Autexier, Christoph Benzmüller, Armin Fiedler, Helmut Horacek, and Bao Quoc Vo. Assertion-level proof representation with under-specification. *Electronic in Theoretical Computer Science*, 93:5–23, 2004.

[6] Christoph Benzmüller. SFB 378 project: Dialog. http://www.ags.uni-sb.de/~chris/dialog/.

[7] Christoph Benzmüller, Armin Fiedler, Malte Gabsdil, Helmut Horacek, Ivana Kruijff-Korbayova, Manfred Pinkal, Jörg Siekmann, Dimitra Tsovaltzi, Bao Quoc Vo, and Magdalena Wolska. A wizard-of-oz experiment for tutorial dialogues in mathematics. In *Proceedings of the AIED Workshop on Advanced Technologies for Mathematics Education*, pages 471–481, Sydney, Australia, 2003.

[8] Christoph Benzmüller and Volker Sorge. OANTS – an open approach at combining interactive and automated theorem proving. In Manfred Kerber and Michael Kohlhase, editors, *Symbolic Computation and Automated Reasoning*, pages 81–97. A.K.Peters, 2000.

[9] Christoph E. Benzmüller and Bao Quoc Vo. Mathematical domain reasoning tasks in natural language tutorial dialog on proofs. In M. Veloso and

S. Kambhampati, editors, *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, pages 516–522, Pittsburgh, Pennsylvania, USA, 2005. AAAI Press / The MIT Press.

[10] Woodrow Wilson Bledsoe, Robert Stephen Boyer, and William Henry Henneman. Computer proofs of limit theorems. *Artificial Intelligence*, 3:27–60, 1972.

[11] Martin D. S. Braine and Barbara Rumain. Logical Reasoning. In P. H. Mussen, editor, *Handbook of Child Psychology*, volume 3. Wiley, 1983.

[12] Ilja N. Bronstein and Konstantin A. Semendjajew. *Taschenbuch der Mathematik*. Teubner, 1991.

[13] Felix Brosius. *SPSS 8.0 : Professionelle Statistik unter Windows*. International Thomson Publishing, Bonn, 1998.

[14] Alan Bundy. Proof planning. In *AIPS*, pages 261–267, 1996.

[15] Patricia W. Cheng and Keith J. Holyoak. Pragmatic reasoning schemas. *Cognitive Psychology*, 17:391–416, 1985.

[16] Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[17] Armin Fiedler. *P.rex*: An interactive proof explainer. In Rejeev Goré, Alexander Leitsch, and Tobias Nipkow, editors, *Automated Reasoning — 1st International Joint Conference, IJCAR 2001*, number 2083 in LNAI, pages 416–420, Siena, Italy, 2001. Springer Verlag.

[18] Jean Flower, Judith Masthoff, and Gem Stapleton. Generating readable proofs: A heuristic approach to theorem proving with spider diagrams. In *Diagrams '04*, pages 166–181, Cambridge, March 2004. International Conference on the Theory and Application of Diagrams.

[19] Free Software Foundation. Welcome to GNU TeXmacs. http://www.texmacs.org.

[20] Gerhard Gentzen. Untersuchungen über das logische Schliessen. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1934.

[21] Fausto Giunchiglia and Toby Walsh. A theory of abstraction. *Artificial Intelligence*, 1992.

[22] Richard Grandy. *Advanced Logic for Applications*. D. Reidel Publishing Company, Dordrecht, Holland, 1977.

[23] The Omega Group. Welcome to the omega group. http://www.ags.uni-sb.de/~omega/.

[24] Jerry R. Hobbs. Granularity. In *IJCAI*, pages 432–435, 1985.

[25] Joris van der Hoeven. GNU texmacs: A free, structured, wysiwyg and technical text editor. In Daniel Flipo, editor, *Le document au XXI-ième siècle*, volume 39-40, pages 39–50, Metz, 2001. Actes du congrès GUTenberg.

[26] Xiarong Huang. *Human Oriented Proof Presentation: A Reconstructive Approach.* Phd thesis, Universität des Saarlandes, Saarbrücken, Germany, 1994. Published by *infix*, St. Augustin, Germany, Dissertationen zur Künstlichen Intelligenz, Volume 112, 1996.

[27] Philip N. Johnson-Laird. Rules and illusions: A critical study of Rips's "The Psychology of Proof".

[28] Philip N. Johnson-Laird. An end to the controversy? A reply to Rips. *Minds and Machines*, Volume 7(3):425–432, 1997.

[29] Philip N. Johnson-Laird. Deductive reasoning. *Annual Review of Psychology*, 50:109–135, 1999.

[30] Philip N. Johnson-Laird and Ruth M. J. Byrne, editors. *Deduction.* Erlbaum, Hillsdale, NJ, 1991.

[31] Jung Hee Kim and Michael Glass. Evaluating Dialogue Schemata with the Wizard of Oz Computer-Assisted Algebra Tutor. In Lester et al. [33], pages 358–367.

[32] Kenneth R. Koedinger and Ben A. MacLaren. Developing a pedagogical domain theory of early algebra problem solving. Technical Report 02-100, CMU-HCII Tech Repor, 2002.

[33] James C. Lester, Rosa Maria Vicari, and Fábio Paraguaçu, editors. *Intelligent Tutoring Systems, 7th International Conference, ITS 2004, Maceiò, Alagoas, Brazil, August 30 - September 3, 2004, Proceedings*, volume 3220 of *Lecture Notes in Computer Science.* Springer, 2004.

[34] David Maulsby, Saul Greenberg, and Richard Mander. Prototyping an intelligent agent through wizard of oz. In *ACM Conference on Human Factors in Computing Systems, INTERCHI '93.* ACM Press, 1993.

[35] Gordon McCalla, Jim Greer, Bryce Barrie, and Paul Pospisel. Granularity hierarchies. In Fritz Lehmann, editor, *Semantic Networks in Artificial Intelligence*, pages 363–375. Pergamon Press, Oxford, 1992.

[36] George A. Miller. The magical number seven plus or minus two - some limits on our capacity for processing information. *Psychological Review*, 63:81–97, 1956.

[37] Neil V. Murray. Completely non-clausal theorem proving. *Artificial Intelligence*, 18:67–85, 1982.

[38] Sylvie Normand-Assadi, Lalina Coulange, Elisabeth Delozanne, and Brigitte Grugeon. Linguistic Markers to Improve the Assessment of Students in Mathematics: An Exploratory Study. In Lester et al. [33], pages 380–389.

[39] Willard V. O. Quine. *Methods of Logic*. Holt, Rinehart & Winston, 3 edition, 1972.

[40] Lance J. Rips. *The psychology of proof : deductive reasoning in human thinking*. MIT Press, Cambridge, MA, 1994.

[41] Lance J. Rips. Goals for a theory of deduction: Reply to Johnson-Laird. *Minds and Machines*, 7(3):409–424, 1997.

[42] Neil Robertson, Daniel Sanders, Paul Seymour, and Robin Thomas. A new proof of the four colour theorem. *Electronic Research Announcements of the American Mathematical Society*, 2:17–25, 1996.

[43] Carolyn Penstein Rosé, Johanna D. Moore, Kurt VanLehn, and David Albritton. A comparative evaluation of socratic versus didactic tutoring. In *23rd Annual Conference of the Cognitive Science Society*, Edinburgh, Scotland, August 2001.

[44] Jörg Siekmann, Christoph Benzmüller, and Serge Autexier. Computer supported mathematics with omega. *Journal of Applied Logic*, 2005. To appear.

[45] Albert Thoralf Skolem. *On mathematical logic*. Harvard University Press, 1967.

[46] David Tall. Cognitive aspects of proof, with special reference to the irrationality of $\sqrt{2}$. In *Proceedings of the Third International Conference for the Psychology of Mathematics Education*, pages 206–207, Warwick, 1979.

[47] Hao Wang. Toward mechanical mathematics. *IBM Journal for Research and Development*, 4:2–22, 1960.

[48] Claus-Peter Wirth. Full first-order sequent and tableau calculi with preservation of solutions and the liberalized delta-rule but without skolemization. In *FTP (LNCS Selection)*, pages 282–297, 1998.

Appendix A

Instruction Documents from the 2005 Wizard-of-Oz Experiment

A.1 Teilnehmer Informationen zum Experiment

1 Ziel des Experiments:

- Sie sollen in einer Brauchbarkeitsstudie die α-Version des DIALOG-Beweis-Tutor-Systems testen.
- Also: Nicht Sie werden getestet, sondern unser System.
- Am Ende des Experiments werden wir Sie in einem Fragebogen nach Ihrer Kritik zum System befragen.

2 Verhalten während des Experiments

- Unser System ist konzipiert als ein interaktives Beweis-Tutor-System, welches Ihnen beim schrittweisen Beweisen von mathematischen Theoremen Feedback zu Ihren einzelnen Beweisschritten liefert. Deshalb: Gehen Sie beim Beweisen möglichst schrittweise vor und vermeiden Sie die Eingabe längerer Argumentationsketten. Dies zwingt das System, möglichst häufig auf Ihre Eingaben zu antworten.
- Unser System versteht einen Mix aus natürlicher Sprache (Deutsch) und Formeln, wie Sie es üblicherweise aus dem Studium oder dem Mathematikunterricht kennen. Machen Sie davon Gebrauch. Die Eingabe zum System ist aber noch reduziert auf *getippte Sprache/Formeln*; gesprochene Sprache wird noch nicht unterstützt. Benutzen Sie bitte keine Umlaute, diese werden nicht erkannt.
- Falls Sie mit LaTeX vertraut sind, können Sie mathematische Zeichen auch mit \ und dem gewohnten Symbolnamen erzeugen, beispielsweise \subset. Danach <return> drücken, und das Symbol erscheint.
- Copy & Paste: Markieren mit der linken Maustaste. Einfügen mit der mittleren Taste.
- Sie dürfen sich Notizen (z.B. Kritik zum System) während des Experiments auf einem Blatt Papier machen.
- Sie dürfen Ihre Kritik und Kommentare gerne auch laut äussern; sie sollen auch nach Möglichkeit laut *denken*. Wir werden diese Informationen aufzeichnen und ebenfalls auswerten.
- Wichtig ist aber: Beweisrelevante Information sollte möglichst über das Interface zum DIALOG-Beweis-Tutor-System eingegeben werden, weil das DIALOG-Beweis-Tutor-System die gesprochene Sprache ja leider noch nicht versteht.

3 Nach dem Experiment

- Sie sollen uns in einem Interview/Fragebogen möglichst umfangreiche Kommentare und Kritik zum System liefern.

A.2 Ablauf und Instruktionen zum Experiment

Christoph Benzmüller und Marvin Schiller

Hinweise:

- Dieses Dokument gibt den Ablauf der gesamten Experiment vor. Eine strikte Einhaltung der Reihenfolge ist zwingend erforderlich. Bitte kontaktieren Sie im Fall von Abweichungen den Experimentator.
- Geben Sie Ihr Bestes! Je nach Abschneiden winkt ein kleines Bonuspräsent.
- Die Gesamtzeit des Experiments beträgt ca. 115min.

1 Ausfüllen des Teils I des Fragebogens [ca. 5min]

Vor Beginn des Experiments werden bzw. wurden Sie gebeten, Teil I unseres Fragebogens auszufüllen. Vergewissern Sie sich, da dies erfolgt ist, bevor Sie mit dem Experiment fortfahren.

Der Fragebogen wird ihnen auf einem Fenster des Dialogsystems angezeigt. Alternativ können Sie auch einen Papierbogen erhalten. Gegebenenfalls kontaktieren Sie den Experimentator und sprechen ihn auf den Fragebogen an.

2 Einarbeitung in die behandelte Domäne [ca. 25min]

Nehmen Sie sich ca. 25min Zeit zur Einarbeitung in die im Experiment behandelte mathematische Domäne *Mengen und Relationen*. Dazu wurde Ihnen das 3-seitige Dokument *"Mengen und Relationen – Eine Auffrischung"* ausgehändigt. Das Experiment wird erst nach dieser Einarbeitungsphase fortgesetzt.

Ein Fenster mit der ,,Auffrischung" steht auch im Interface des Dialogsystems zur Verfügung.

3 Benutzung des DIALOG Systems [ca. 5min]

Sie sollen sich nun mit der graphischen Eingabeoberfläche unseres DIALOG Systems vertraut machen. Ein Hinweisblatt mit Erläuterungen und Tipps zur Oberfläche wurde Ihnen ausgehändigt. Tippen Sie nun den folgenden Text ein und stellen sie gegebenfalls Fragen an den Experimentator:

Die geordneten Paare dienen zur Definition des kartesischen Produktes (Kreuzproduktes) $A \times B$ zweier Mengen, das durch

$$A \times B := \{(a, b)|a \in A \wedge b \in B\}$$

erklärt wird und wieder eine Menge ist.

Drücken Sie nach erfolgter Eingabe auf den ,,absenden" Knopf der Benutzeroberfäche. Danach werden Sie aufgefordert, das Experiment zu beginnen.

4 Zum Warmwerden: Aufgabe W [ca. 15min]

Satz 1. Seien R und S Relationen in einer beliebigen Menge M. Es gilt:

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

Beweis [Führen Sie den Beweis nun interaktiv mit dem System; Sie haben ca. 15min Zeit. Bitte gehen sie möglichst Schrittweise vor; dadurch bieten Sie dem System die Möglichkeit auf Ihre Beweischritte zu reagieren.] □

5 Experiment: Aufgabe A [ca. 25min]

Satz 2. Seien R, S und T Relationen in einer beliebigen Menge M. Es gilt:

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

Beweis [Führen Sie den Beweis nun interaktiv mit dem System; Sie haben ca. 25min Zeit.] □

6 Experiment: Aufgabe B [ca. 25min]

Satz 3. Seien R, S und T Relationen in einer Menge M. Es gilt:

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

Beweis [Führen Sie den Beweis nun interaktiv mit dem System; Sie haben ca. 25min Zeit.] □

7 Bonus-Aufgabe C [falls Zeit vorhanden]

Satz 4. Seien R und S Relationen in einer Menge M. Es gilt:

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

Beweis [Führen Sie den Beweis nun interaktiv mit dem System.] □

8 Ausfüllen von Teil II des Fragebogens [ca. 10min]

Bitte füllen Sie nun Teil II unseres Fragebogens aus. Sobald sich das Dialogsystem beendet hat, erscheint dieser in einem neuen Fenster.

9 Erhalt der Aufwandsentschädigung [ca. 5min]

Nachdem sie den Fragebogen komplett ausgefüllt haben erhalten Sie die vereinbarte Aufwandsentschädigung. Je nach Abschneiden im Test erhalten Sie ein kleines Bonuspräsent.

A.3 Mengen und Relationen – Eine Auffrischung

von Christoph Benzmüller und Marvin Schiller

Hinweis: Mit Ausnahme von Abschnitt 1 ist der folgende Text entnommen aus *I.N. Bronstein and K.A. Semendjajew, Taschenbuch der Mathematik, 25. Auflage, Teubner, 1991.* Einige leichte Modifikationen, meist syntaktische, wurden vorgenommen um die Präsentation und die Symbole besser auf unser DIALOG-System abzustimmen.

1 Erinnerung: Begriffe der Mathematischen Logik

Wir erinnern kurz (informal) an folgende Grundbegriffe und Schreibweisen der Mathematischen Logik; seien A, B Aussagen der mathematischen Logik:

- $(A = B)$ ... A ist gleich B
- $\neg A$ ... A gilt nicht
- $(A \wedge B)$ ... A ist wahr und B ist wahr
- $(A \vee B)$ ... A ist wahr oder B ist wahr, oder beide
- $(A \Rightarrow B)$ ... aus A folgt B, d.h. $A \Rightarrow B$ ist genau dann wahr, wenn $A \wedge B$ oder $\neg A$ wahr ist
- $(A \Leftrightarrow B)$ ... A gilt genau dann wenn B gilt
- $\forall x(A)$ ... für alle x gilt: Aussage A ist wahr
- $\exists x(A)$ ... es gibt ein x, so dass gilt: Aussage A ist wahr

Die Bindungsstärke dieser Operationen ist durch folgende Folge charakterisiert: $=, \neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$. D.h. $=$bindet stärker als $\neg$, dann folgt $\wedge$, usw . Dies ermöglicht es, Klammern zu vermeiden: $A = B \wedge C \Rightarrow D \vee E$ steht für $(((A = B) \wedge C) \Rightarrow (D \vee E))$.

Einige wichtige Sätze der Aussagenlogik sind:

$(A1)$	$A \vee \neg A$	(tertium non datur)
$(A2)$	$\neg(A \wedge \neg A)$	(Widerspruch)
$(A3)$	$\neg\neg A \Leftrightarrow A$	(doppelte Verneinung)
$(A4)$	$\neg(A \wedge B) \Leftrightarrow \neg A \vee \neg B$	(deMorganI)
$(A5)$	$\neg(A \vee B) \Leftrightarrow \neg A \wedge \neg B$	(deMorganII)
$(A6)$	$A \Rightarrow B \Leftrightarrow \neg B \Rightarrow \neg A$	(Kontraposition)
$(A7)$	$(A \Rightarrow B) \wedge A \Rightarrow B$	(modus ponens)
$(A8)$	$(A \Rightarrow B) \wedge \neg B \Rightarrow \neg A$	(modus tollens)
$(A9)$	$(A \Rightarrow B) \wedge (B \Rightarrow C) \Rightarrow (A \Rightarrow C)$	(modus barbara)
$(A10)$	$A \wedge (B \vee C) \Leftrightarrow (A \wedge B) \vee (A \wedge C)$	(Distributivität I)
$(A11)$	$A \vee (B \wedge C) \Leftrightarrow (A \vee B) \wedge (A \vee C)$	(Distributivität II)

Einige wichtige Sätze zum Umgang mit Quantoren sind:

$(P1)$	$\exists x(P(x) \vee Q(x)) \Leftrightarrow \exists x(P(x)) \vee \exists x(Q(x))$
$(P2)$	$\forall x(P(x) \wedge Q(x)) \Leftrightarrow \forall x(P(x)) \wedge \forall x(Q(x))$
$(P3)$	$\neg\exists x(P(x)) \Leftrightarrow \forall x(\neg P(x))$
$(P4)$	$\neg\forall x(P(x)) \Leftrightarrow \exists x(\neg P(x))$
$(P5)$	$\forall x(P(x)) \vee \forall x(Q(x)) \Rightarrow \forall x(P(x) \vee Q(x))$
$(P6)$	$\exists x(P(x) \wedge Q(x)) \Rightarrow \exists x(P(x)) \wedge \exists x(Q(x))$
$(P7)$	$\forall x(P \vee Q(x)) \Leftrightarrow P \vee \forall x(Q(x))$
$(P8)$	$\exists x(P \vee Q(x)) \Leftrightarrow P \vee \exists x(Q(x))$
$(P9)$	$(\forall x(P(x)) \Rightarrow Q) \Leftrightarrow \exists x(P(x) \Rightarrow Q)$
$(P10)$	$(\exists x(P(x)) \Rightarrow Q) \Leftrightarrow \forall x(P(x) \Rightarrow Q)$
$(P11)$	$(P \Rightarrow \forall x(Q(x))) \Leftrightarrow \forall x(P \Rightarrow Q(x))$
$(P12)$	$(P \Rightarrow \exists x(Q(x))) \Leftrightarrow \exists x(P \Rightarrow Q(x))$

2 Grundbegriffe der Mengenlehre

2.1 Mengen und Elemente

Die beiden grundlegenden Begriffe der Mengenlehre sind der Begriff der *Menge* und derjenige der *Element-Beziehung*. Unter einer Menge versteht man eine Zusammenfassung gewisser Dinge zu einem neuen einheitlichen Ganzen; die dabei zusammengefaßten Dinge heißen die *Elemente* der betreffenden Menge. Ist ein Ding a Element einer Menge M, so schreibt man kurz

$$a \in M \quad (\text{gelesen : } a \, \text{ist Element von} \, M);$$

ist a nicht Element von M, so schreibt man dafür $a \notin M$.

Das *Extensionalitätsprinzip* für Mengen besagt, daß zwei Mengen A, B genau dann gleich sind, wenn sie dieselben Elemente enthalten:

$$A = B \Leftrightarrow \forall x(x \in A \Leftrightarrow x \in B).$$

Bei den Anwendungen mengentheoretischer Überlegungen betrachtet man meist eine bestimmte Art von Dingen (z.B reelle Zahlen, Punkte eines Raumes, Funktionen) als gegeben und bildet von ihnen ausgehend Mengen. Derartige Dinge, die dann selbst nicht als Mengen aufgefasst werden, nennt man *Urelemente*.

Eine Menge läßt sich dadurch beschreiben, daß man zwei Eigenschaften angibt, die sämtlichen Elementen dieser Menge, jedoch keinen anderen Dingen zukommt. Ist $H(x)$ eine Eigenschaft von wohldefinierten Dingen, so bezeichnet man die Menge aller Dinge a mit der Eigenschaft $H(a)$ durch $\{x|H(x)\}$. Also gilt stets

$$a \in \{x|H(x)\} \Leftrightarrow H(a) \quad (\text{falls die Variable} \, a \, \text{nicht in dem Ausdruck} \, H(x) \, \text{vorkommt}).$$

Da Mengen und Urelemente als Elemente von Mengen auftreten können, bezeichnet man beide Arten von Dingen als Elemente schlechthin.

2.2 Teilmengen

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 $A \subseteq B$. Gilt außer $A \subseteq B$ auch $A \neq B$, so schreibt man $A \subset B$ und nennt A eine *echte Teilmenge* von B. Die solcherart zwischen Mengen erklärten Beziehungen $\subseteq, \subset$ werden *Inklusion* bzw. *echte Inklusion* genannt.

Ein wichtiger Zusammenhang zwischen Inklusion und Gleichheit von Mengen folgt aus dem Extensionalitätsprinzip (2.1): Für Mengen A, B gilt

$$A = B \Leftrightarrow A \subseteq B \wedge B \subseteq A$$

Aus dieser Formel entnimmt man ein oft benutztes *Beweisprinzip*: Um die Gleichheit zweier Mengen A, B zu beweisen, hat man die beiden Inklusionen $A \subseteq B$ und $B \subseteq A$ zu beweisen.

2.3 Spezielle Mengenbildungsprinzipien

Aus der in 2.1 genannten Art der Bildung von Mengen $\{x|H(x)\}$ ergeben sich einige wichtige Spezialfälle:

(1) Wählt man als Eigenschaft $H(x)$ in 2.1 speziell irgendeine Eigenschaft, die auf kein Ding zutrifft, etwa $x \neq x$, so gilt für kein Ding a, daß $a \in \{x|H(x)\}$, also $\forall a(a \notin \{x|H(x)\})$. Da es wegen des Extensionalitätsprinzips nur eine Menge ohne Elemente geben kann, nennt man diese spezielle Menge die *leere Menge*; man bezeichnet sie mit $\emptyset$:

$$\emptyset := \{x|x \neq x\}.$$

(2) Ist M eine Menge und $H(x)$ eine für die Elemente von M erklärte Eigenschaft, so kann man immer die Menge $\{x|x \in M \wedge H(x)\}$ bilden:

$$\text{Stets ist} \{x \in M|H(x)\} := \{x|x \in M \wedge H(x)\} \,\text{und} \{x \in M|H(x)\} \subseteq M.$$

(3) Zu jeder Menge M kann man die Menge aller Teilmengen von M, die sogenannte *Potenzmenge* $\wp M$ von M, bilden:

$$\wp M := \{V|V \subseteq M\}.$$

Es ist stets $\emptyset \in \wp M$ und $M \in \wp M$.

3 Operationen mit Mengen und Mengensystemen

3.1 Vereinigung und Durchschnitt von Mengen

Zu vorgegebenen Mengen A, B bilden wir die *Vereinigung* (*Vereinigungsmenge*) $A \cup B$ und den *Durchschnitt* (die *Durchschnittsmenge*) $A \cap B$ von A, B durch Festsetzungen

$$\begin{aligned} A \cup B &:= \{x | x \in A \vee x \in B\}, \\ A \cap B &:= \{x | x \in A \wedge x \in B\}. \end{aligned}$$

3.2 Differenz, symmetrische Differenz und Komplement von Mengen

Für Mengen A, B, E, für die noch $A \subseteq E$ gilt, werden die *Differenz* $A \setminus B$ von A, B, die *symmetrische Differenz* $A \div B$ von A, B und das *Komplement* $\mathcal{C}_E A$ von A bezüglich E erklärt:

$$\begin{aligned} A \setminus B &:= \{x | x \in A \wedge x \notin B\}, \\ A \div B &:= (A \setminus B) \cup (B \setminus A), \\ \mathcal{C}_E A &:= (E \setminus A). \end{aligned}$$

3.3 Kartesisches Produkt von Mengen

Unter dem *geordneten Paar* (a, b) zweier Elemente a, b versteht man die Menge

$$(a, b) := \{\{a\}, \{a, b\}\}.$$

Es gilt für beliebige Elemente a, b, c, d

$$(a, b) = (c, d) \Leftrightarrow a = c \wedge b = d,$$

d.h., beim geordneten Paar (a,b) wird die Reihenfolge der Elemente a, b ausgezeichnet: denn ist $a \neq b$, so $(a, b) \neq (b, a)$. Im Gegensatz dazu gilt für die Zweiermenge[1] $\{a, b\}$ z.B. $\{a, b\} = \{b, a\}$.

Statt ...

Die geordneten Paare dienen zur Definition des *kartesischen Produktes (Kreuzproduktes)* $A \times B$ zweier Mengen, das durch

$$A \times B := \{(a, b) | a \in A \wedge b \in B\}$$

erklärt wird und wieder eine Menge ist.

[1] zweielementige Menge

4 Relationen, Funktionen, Operationen

4.1 Relationen

Anschaulich versteht man unter einer *binären Relation* R in einer Menge M eine Beziehung, die zwischen zwei Elementen $a, b \in M$ entweder besteht oder nicht besteht. Die formale Fassung diesen Begriffs geschieht in folgender Weise: Eine binäre Relation R in einer Menge M ist eine Teilmenge von $M \times M$.

Ist $(a, b) \in M \times M$, so sagt man im Falle, daß $(a, b) \in R$ gilt, daß die Relation R auf die Elemente a, b zutreffe bzw. zwischen ihnen bestehe, und schreibt Rab (alternativ aRb). Im Falle $(a, b) \notin R$ dagegen sagt man, daß die Relation R auf die Elemente a, b nicht zutreffe bzw. zwischen ihnen nicht bestehe.

Das Relationenprodukt zweier binärer Relationen R, S in einer Menge M ist definiert durch

$$R \circ S := \{(x, y) | \exists z (z \in M \wedge (x, z) \in R \wedge (z, y) \in S)\};$$

die *inverse (konverse, reziproke, duale) Relation* von R ist die Relation

$$R^{-1} := \{(x, y) | (y, x) \in R\}.$$

Wichtige Eigenschaften einer binären Relation R in einer Menge M:[2]

Eigenschaft	**charakterisierende Bedingung**
Reflexivität	$(\forall)$ aRa
Irreflexivität	$(\forall) \neg$ aRa
Transitivität	$(\forall)$ aRb $\wedge$ bRc $\Rightarrow$ aRc
Drittengleichheit	$(\forall)$ aRc $\wedge$ bRc $\Rightarrow$ aRb
Symmetrie	$(\forall)$ aRb $\Rightarrow$ bRa
Antisymmetrie	$(\forall)$ aRb $\wedge$ bRa $\Rightarrow a = b$
Asymmetrie	$(\forall)$ aRb $\Rightarrow \neg$ bRa
Linearität	$(\forall)$ aRb $\vee a = b \vee$ bRa
Konnexität	$(\forall)$ aRb $\vee$ bRa
Trichotomie	....

[2] $(\forall)$ bedeutet dabei die auf M beschränkte Quantifizierung der nachfolgenden Variablen a, b, c mittels $\forall$, z.B. im Falle der Symmetrie: $\forall a, b \in M$(aRb $\Rightarrow$ bRa).

A.4 Mengen und Relationen – Eine Auffrischung(2)

Christoph Benzmüller und Marvin Schiller

Hinweis: Diese Einführung basiert auf modifizierten Texten aus *I.N. Bronstein and K.A. Semendjajew, Taschenbuch der Mathematik, 25. Auflage, Teubner, 1991.*

1 Erinnerung: Begriffe der Mathematischen Logik

Wir erinnern kurz (informal) an folgende Grundbegriffe und Schreibweisen der Mathematischen Logik; seien A, B Aussagen der Mathematischen Logik:

- $(A = B)$ ... A ist gleich B
- $\neg A$ ... A gilt nicht
- $(A \wedge B)$ ... A ist wahr und B ist wahr
- $(A \vee B)$ ... A ist wahr oder B ist wahr, oder beide
- $(A \Rightarrow B)$ ... aus A folgt B, d.h. $A \Rightarrow B$ ist genau dann wahr, wenn $A \wedge B$ oder $\neg A$ wahr ist
- $(A \Leftrightarrow B)$ ... A gilt genau dann wenn B gilt
- $\forall x(A)$ ... für alle x gilt: Aussage A ist wahr
- $\exists x(A)$ ... es gibt ein x, so dass gilt: Aussage A ist wahr

Die Bindungsstärke dieser Operationen ist durch folgende Folge charakterisiert: $=, \neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$. D.h. $=$bindet stärker als $\neg$, dann folgt $\wedge$, usw . Dies ermöglicht es, Klammern zu vermeiden: $A = B \wedge C \Rightarrow D \vee E$ steht für $(((A = B) \wedge C) \Rightarrow (D \vee E))$.

2 Grundbegriffe der Mengenlehre

2.1 Mengen und Elemente

Die beiden grundlegenden Begriffe der Mengenlehre sind der Begriff der *Menge* und derjenige der *Element-Beziehung*. Unter einer Menge versteht man eine Zusammenfassung gewisser Dinge zu einem neuen einheitlichen Ganzen; die dabei zusammengefaßten Dinge heißen die *Elemente* der betreffenden Menge. Ist ein Ding a Element einer Menge M, so schreibt man kurz

$$a \in M \quad (\text{gelesen}: a\,\text{ist Element von}\,M);$$

ist a nicht Element von M, so schreibt man dafür $a \notin M$.

Das *Extensionalitätsprinzip* für Mengen besagt, daß zwei Mengen A, B genau dann gleich sind, wenn sie dieselben Elemente enthalten, d.h. $A = B$ genau dann wenn alle x aus A auch Element von B sind und umgekehrt.

Bei den Anwendungen mengentheoretischer Überlegungen betrachtet man meist eine bestimmte Art von Dingen (z.B reelle Zahlen, Punkte eines Raumes, Funktionen) als gegeben und bildet von ihnen ausgehend Mengen. Derartige Dinge, die dann selbst nicht als Mengen aufgefasst werden, nennt man *Urelemente*.

2.2 Teilmengen

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 $A \subseteq B$. Gilt außer $A \subseteq B$ auch $A \neq B$, so schreibt man $A \subset B$ und nennt A eine *echte Teilmenge* von B. Die solcherart zwischen Mengen erklärten Beziehungen $\subseteq, \subset$ werden *Inklusion* bzw. *echte Inklusion* genannt.

Ein wichtiger Zusammenhang zwischen Inklusion und Gleichheit von Mengen folgt aus dem Extensionalitätsprinzip (2.1): Für Mengen A, B gilt $A = B$ genau dann wenn $A \subseteq B$ und $B \subseteq A$.

Aus dieser Formel entnimmt man ein oft benutztes *Beweisprinzip*: Um die Gleichheit zweier Mengen A, B zu beweisen, hat man die beiden Inklusionen $A \subseteq B$ und $B \subseteq A$ zu beweisen.

2.3 Spezielle Mengenbildungsprinzipien

Wir definieren einige spezielle Mengen:

(1) Die leere Menge enthält keine Elemente. Wir bezeichnen sie mit $\emptyset$.

Formal können wir die leere Menge wie folgt definieren: $\emptyset := \{x | x \neq x\}$.

(2) Zu jeder Menge M kann man die Menge aller Teilmengen von M, die sogenannte *Potenzmenge* $\wp M$ von M, bilden: $\wp M := \{V | V \subseteq M\}$. Es ist stets $\emptyset \in \wp M$ und $M \in \wp M$.

3 Operationen mit Mengen und Mengensystemen

3.1 Vereinigung und Durchschnitt von Mengen

Zu vorgegebenen Mengen A, B bilden wir die *Vereinigung* (*Vereinigungsmenge*) $A \cup B$ und den *Durchschnitt* (die *Durchschnittsmenge*) $A \cap B$ von A, B. Ein Objekt x ist Element der Vereinigungsmenge $A \cup B$, genau dann wenn x Element von A ist oder x Element von B ist. x ist Element der Durchschnittsmenge $A \cap B$, genau dann wenn x Element von A ist und x Element von B ist.

3.2 Differenz, symmetrische Differenz und Komplement von Mengen

Für Mengen A, B, E, für die noch $A \subseteq E$ gilt, werden die *Differenz* $A \setminus B$ von A, B, die *symmetrische Differenz* $A \div B$ von A, B und das *Komplement* $\mathcal{C}_E A$ von A bezüglich E erklärt. Ein Objekt x ist Element der Differenzmenge $A \setminus B$, genau dann wenn x Element von A ist und x nicht Element von B ist. x ist Element der *symmetrischen Differenz* $A \div B$, genau dann wenn x Element der Menge $(A \setminus B) \cup (B \setminus A)$ ist. x ist Element des *Komplements* $\mathcal{C}_E A$ von A bezüglich E, genau dann wenn x Element der Menge $(E \setminus A)$ ist.

3.3 Kartesisches Produkt von Mengen

Unter dem *geordneten Paar* (a, b) zweier Elemente a, b versteht man die Menge {{a}, {a,b}} .

Für beliebige Elemente a, b, c, d gilt: $(a, b) = (c, d)$ genau dann wenn $a = c$ und $b = d$.

Die geordneten Paare dienen zur Definition des *kartesischen Produktes (Kreuzproduktes)* $A \times B$ zweier Mengen. Ein Paar (a, b) ist Element des Kreuzprodukts $A \times B$, genau dann wenn $a \in A$ und $b \in B$.

4 Relationen, Funktionen, Operationen

4.1 Relationen

Anschaulich versteht man unter einer *binären Relation* R in einer Menge M eine Beziehung, die zwischen zwei Elementen $a, b \in M$ entweder besteht oder nicht besteht. Die formale Fassung diesen Begriffs geschieht in folgender Weise: Eine binäre Relation R in einer Menge M ist eine Teilmenge von $M \times M$.

Ist $(a, b) \in M \times M$, so sagt man im Falle, daß $(a, b) \in R$ gilt, daß die Relation R auf die Elemente a, b zutreffe bzw. zwischen ihnen bestehe, und schreibt Rab (alternativ aRb). Im Falle $(a, b) \notin R$ dagegen sagt man, daß die Relation R auf die Elemente a, b nicht zutreffe bzw. zwischen ihnen nicht bestehe.

Das Relationenprodukt zweier binärer Relationen R, S in einer Menge M ist wie folgt definiert: Sei $x, y \in M$. Ein Paar (x, y) ist Element von $R \circ S$, genau dann wenn es ein z in M gibt, so dass $(x, z) \in R$ und $(z, y) \in S$.

Die *inverse (konverse, reziproke, duale) Relation* von R bezeichnen wir durch R^{-1}. Sie ist wie folgt definiert: Ein Paar (x, y) ist Element von R^{-1}, genau dann wenn $(y, x) \in R$.

4.2 Einige wichtige Eigenschaften binärer Relationen

Eine binäre Relation R in einer Menge M wird als *reflexiv* bezeichnet, wenn für jedes Element $a \in M$ das Paar (a, a) in R enthalten ist.

Eine binäre Relation R in einer Menge M wird als *transitiv* bezeichnet, wenn für alle Elemente $a, b, c \in M$ gilt: Aus $(a, b) \in R$ und $(b, c) \in R$ folgt stets $(a, c) \in R$.

Eine binäre Relation R in einer Menge M wird als *symmetrisch* bezeichnet, wenn für alle Elemente $a, b \in M$ gilt: Wenn $(a, b) \in R$, dann folgt stets $(b, a) \in R$.

Eine binäre Relation R in einer Menge M wird als *asymmetrisch* bezeichnet, wenn für alle Elemente $a, b \in M$ gilt: Wenn $(a, b) \in R$, dann folgt stets, daß nicht $(b, a) \in R$ gilt.

Beispiel-Beweis [3]

Theorem 1. *Sei R eine Relation in einer Menge M. Es gilt:*

$$R = (R^{-1})^{-1}$$

Proof. 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. □

[3] Autor: Christoph Benzmüller

A.5 Benutzung des Tutorsystem-Interfaces

Interaktion

Dem Tutorsystem können über das Interface Nachrichten geschickt werden, die nach einer gewissen Zeit beantwortet werden. Beachten Sie:

- Es kann immer nur eine Nachricht auf einmal geschickt werden. Senden Sie die nächste Nachricht erst, wenn sie eine Antwort erhalten haben (dies kann etwas Zeit in Anspruch nehmen). Solange noch keine Antwort erfolgt ist, ist kein Versenden einer weiteren Nachricht möglich - Sie können aber schon an der nächsten Nachricht schreiben.

- Schicken Sie Ihre Nachrichten immer schrittweise, d.h. fuer jeden Beweisschritt oder jeden Beitrag/jede Frage genau eine Nachricht. Dies ermöglicht eine gezielte Antwort.

- Zum Abschicken der Nachrichten benutzen Sie den "Abschicken"-Knopf in der unteren Menueleiste. Die Enter-Taste dagegen dient nicht zum Abschicken, sondern zum Einfügen eines Zeilenumbruchs.

Mathematische Zeichen

In das Textfenster können mathematischer Text, Symbole und Formeln auf unterschiedliche Weise eingegeben werden. Symbole erhält man wie folgt:

- Befindet sich der Cursor in der Eingabezeile, so erscheint ein Menü mit unterschiedlichen Symbolen. Per Mausclick fügt man das Symbol an der Cursorposition ein.

- Symbole koennen mit dem Latex-Symbolnamen eingefügt werden. Dazu Backslash drücken und den Namen eingeben, bspw. `\subset` .

- Symbole können mit deutschen Symbolnamen eingefügt werden. Dazu Backslash drücken und den Namen gemäss der Tabelle eingeben, bspw. `\teilmenge` .

Zeichen	Latex-Befehl	deutscher Befehl	Schnelltaste
$($	(	\runde-klammer-links	(
$)$	)	\runde-klammer-rechts	)
$\{$	\lbrace	\gesch-klammer-links	{
$\}$	\rbrace	\gesch-klammer-rechts	}
$\vert$	|	\strich	
$\cup$	\cup	\vereinigung	
$\cap$	\cap	\schnittmenge	
$\setminus$	\setminus	\differenz	
$\times$	\times	\produkt	
$\subset$	\subset	\echte-teilmenge	
$\supset$	\supset	\echte-obermenge	
$\subseteq$	\subseteq	\teilmenge	
$\supseteq$	\supseteq	\obermenge	
$=$	=	\gleich	
$\neq$	\neq	\ungleich	
$\in$	\in	\element	
$\notin$	\notin	\kein-element	
$\circ$	\circ	\verknuepft	
$^{-1}$	^-1	\invers	
$\exists$	\exists	\existiert	
$\nexists$	\nexists	\existiert-kein	
$\forall$	\forall	\fueralle	
$\neg$	\neg	\nicht	
$\vee$	\vee	\oder	
$\wedge$	\wedge	\und	
$\Rightarrow$	\Rightarrow	\impliziert	
$\Leftarrow$	\Leftarrow	\folgt-aus	
$\Leftrightarrow$	\Leftrightarrow	\aequivalent	
$\{ \vert \}$	\lbrace | \rbrace	\menge	
$(,)$	(,)	\paar	
$\emptyset$	\emptyset	\leeremenge	

Table A.1:

Machen Sie sich nun mit dem Interface vertraut! Versuchen Sie nun, folgende Textstücke in das Interface zu schreiben. Wenn ihnen dies gelungen ist, schicken Sie den Text ab.

Durch Markieren mit der linken Maustaste und Einfügen mit der mittleren Maustaste können Textstücke kopiert werden (copy&paste). Probieren Sie dies nach Möglichkeit aus.

Falls Sie Hilfe benötigen, wenden Sie sich bitte an Ihren Versuchsleiter!

`Jede Relation` R `besitzt eine inverse Relation` R^{-1}`.`

Appendix B

Master Solutions for the Experiment Exercises

B.1 Master Solutions for Exercise W

Tutor 1

z.z.: $(R \circ S)^{-1} = S^{-1} \circ R^{-1}$

$$\begin{array}{lll}
\text{Bew.: } \subseteq) & \text{Sei } (x,y) \in (R \circ S)^{-1} \qquad x,y \in M \\
& \Rightarrow (y,x) \in R \circ S \\
& \Rightarrow \exists z \in M : (y,z) \in R \wedge (z,x) \in S \\
& \Rightarrow \exists z : (z,y) \in R^{-1} \wedge (x,z) \in S^{-1} \\
& \Rightarrow (x,y) \in S^{-1} \circ R^{-1} \\
\supseteq) & \text{einfach rückwärts, da alle Implikationen Äquivalenzen sind. } \#
\end{array}$$

Tutor 2

$$\begin{array}{ll}
\text{Bew: ,,}\subseteq\text{'':} & \text{Sei } (a,b) \in (R \circ S)^{-1} \\
& \Rightarrow (b,a) \in R \circ S \\
& \Rightarrow \exists z \in M : (b,z) \in R \text{ und } (z,a) \in S. \text{ Wähle solch ein } z. \\
& \Rightarrow (z,b) \in R^{-1} \text{ und } (a,z) \in S^{-1} \\
& \Rightarrow (b,a) \in S^{-1} \circ R^{-1}. \\
\\
\text{,,}\supseteq\text{'':} & \text{Sei nun umgekehrt } (a,b) \in S^{-1} \circ R^{-1} \\
& \exists z \in M : (a,z) \in S^{-1} \text{ und } (z,b) \in R^{-1} \\
& \Rightarrow (z,a) \in S \text{ und } (b,z) \in R \\
& \Rightarrow (b,a) \in R \circ S
\end{array}$$

Damit ist $(R \circ S)^{-1} \subseteq S^{-1} \circ R^{-1}$ und $S^{-1} \circ R^{-1} \subseteq (R \circ S)^{-1}$.

$\Rightarrow$ Behauptung.

Tutor 3

Bew:

$$\begin{aligned}(R \circ S)^{-1} = \{(y,x)|(x,y) \in (R \circ S)\} &= \{(y,x)|\exists z[z \in M \wedge (x,z) \in R \wedge (z,y) \in S]\} \\ &= \{(y,x)|\exists z[z \in M \wedge (z,x) \in R^{-1} \wedge (y,z) \in S^{-1}]\} \\ &= \{(y,x)|\exists z[z \in M \wedge (y,z) \in S^{-1} \wedge (z,x) \in R^{-1}]\} \\ &= S^{-1} \circ R^{-1}\end{aligned}$$

Tutor 4

$(x,y) \in (R \circ S)^{-1}$
$(y,x) \in (R \circ S)$
$(y,z) \in R \wedge (z,x) \in S$
$(z,y) \in R^{-1} \wedge (x,z) \in S^{-1}$
$(x,z) \in S^{-1} \wedge (z,y) \in R^{-1}$
$(x,y) \in S^{-1} \circ R^{-1}$

B.2 Master Solutions for Exercise A

Tutor 1

z.z.: $(R \cup S) \circ T = (R \circ T) \cup (S \circ T)$

Bew.: $\subseteq$) Sei $(x,y) \in (R \cup S) \circ T x, y \in M$
$\Rightarrow \exists z \in M : (x,z) \in (R \cup S) \wedge (z,y) \in T$
$\Rightarrow \exists z \in M : ((x,z) \in R \wedge (z,y) \in T) \vee ((x,z) \in S \wedge (z,y) \in T)$
$\Rightarrow ((x,y) \in R \circ T) \vee ((x,y) \in S \circ T)$
$\Rightarrow (x,y) \in (R \circ T) \cup (S \circ T)$

$\supseteq$) wieder rückwärts #

Tutor 2

Bew: „$\subseteq$": Sei $(a,b) \in (R \cup S) \circ T$
$\Rightarrow \exists z \in M : (a,z) \in R \cup S$ und $(z,b) \in T$
$\Rightarrow (a,z) \in R$ oder $(a,z) \in S$.

1. Fall: $(a,z) \in R$ (2. Fall analog)
$\Rightarrow (a,b) \in R \circ T \subseteq (R \circ T) \cup (S \circ T)$

„$\supseteq$": Sei $(a,b) \in (R \circ T) \cup (S \circ T)$.
$\Rightarrow (a,b) \in R \circ T$ oder $(a,b) \in S \circ T$

1. Fall $(a,b) \in R \circ T$ (2. Fall analog)
$\Rightarrow \exists z \in M : (a,z) \in R$ und $(z,b) \in T$
$\Rightarrow (a,z) \in R \subseteq R \cup S$
$\Rightarrow (a,z) \in (R \cup S) \circ T$
$\Rightarrow$ Behauptung.

Tutor 3

$$\begin{aligned}
(R \cup S) \circ T &= \{(x,y) | \exists z[z \in M \wedge (x,z) \in (R \cup S) \wedge (z,y) \in T]\} \\
&= \{(x,y) | \exists z[z \in M \wedge (z,y) \in T \wedge [(x,z) \in R \\
&\vee (x,z) \in S]]\} \\
&=^{\text{Distributivität}} \{(x,y) | \exists z(z \in M \wedge (z,y) \in T \wedge (x,z) \in R) \\
&\vee \exists z(z \in M \wedge (z,y) \in T \wedge (x,z) \in S)\} \\
&= \{(x,y) | \exists z(z \in M \wedge (x,z) \in R \wedge (z,y) \in T\} \\
&\cup \{(x,y) | \exists z(z \in M \wedge (x,z) \in S \wedge (z,y) \in T\} \\
&= (R \circ T) \cup (S \circ T)
\end{aligned}$$

Tutor 4

$(x,y) \in (R \cup S) \circ T$
$(x,z) \in (R \cup S) \wedge (z,y) \in T$
$((x,z) \in R \vee (x,z) \in S) \wedge (z,y) \in T$
$((x,z) \in R \wedge (z,y) \in T) \vee ((x,z) \in S \wedge (z,y) \in T)$
$(x,y) \in R \circ T \vee (x,y) \in S \circ T$
$(x,y) \in (R \circ T) \cup (S \circ T) \quad \square$

B.3 Master Solutions for Exercise B

Tutor 1

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

$$\text{Bew.: } (R \cup S) \circ T =_A (R \circ T) \cup (S \circ T) \quad \begin{aligned} &=_{inv.Rel.} ((R \circ T)^{-1})^{-1} \cup ((S \circ T)^{-1})^{-1} \\ &= (T^{-1} \circ R^{-1})^{-1} \cup (T^{-1} \circ S^{-1})^{-1} \ \# \end{aligned}$$

Tutor 2

$$\begin{aligned} (T^{-1} \circ S^{-1})^{-1} \cup (T^{-1} \circ R^{-1})^{-1} \quad &=_W (S^{-1})^{-1} \circ (T^{-1})^{-1} \cup (R^{-1})^{-1} \circ (T^{-1})^{-1} \\ &= S \circ T \cup R \circ T \\ &=_A (S \cup R) \circ T \\ &= (R \cup S) \circ T. \end{aligned}$$

Tutor 3

$$\begin{aligned} (T^{-1} \circ S^{-1})^{-1} \cup (T^{-1} \circ R^{-1})^{-1} &=_{\text{Aufgabe W}} (S^{-1})^{-1} \circ (T^{-1})^{-1} \cup (R^{-1})^{-1} \circ (T^{-1})^{-1} \\ &=_{\text{zus. Beh.}} (S \circ T) \cup (R \circ T) \\ &= (R \circ T) \cup (S \circ T) \\ &=_{\text{Aufgabe A}} (R \cup S) \circ T \end{aligned}$$

Zusatz-Behauptung: $(R^{-1})^{-1} = R$
Bew:

$$\begin{aligned} \left(R^{-1}\right)^{-1} &= [\{(x,y)|(y,x) \in R\}]^{-1} \\ &= \{(y,x)|(y,x) \in R\} \\ &= R \end{aligned}$$

Tutor 4

$(x, y) \in (R \cup S) \circ T$
$(x, y) \in (((R \cup S) \circ T)^{-1})^{-1}$
$(y, z) \in ((R \cup S) \circ T)^{-1}$
$(y, x) \in T^{-1} \circ (R \cup S)^{-1}$
$(y, z) \in T^{-1} \wedge (z, x) \in (R \cup S)^{-1}$
$(y, z) \in T^{-1} \wedge ((z, x) \in R^{-1} \vee (z, x) \in S^{-1})$
$(y, x) \in T^{-1} \circ R^{-1} \vee (y, x) \in T^{-1} \circ S^{-1}$

$(x, y) \in (T^{-1} \circ R^{-1})^{-1} \vee (x, y) \in (T^{-1} \circ S^{-1})^{-1}$
$(x, y) \in (T^{-1} \circ R^{-1})^{-1} \cup (T^{-1} \circ R^{-1})^{-1} \quad \square$

Appendix C

Example Log-File

Time: 20050720-14:27:51
Student: (...)
Tutor: (...)
(0 "200705142751" "tutor" "Machen Sie Sich mit dem Interface vertraut (Nachrichten mit dem \"absenden\" Knopf verschicken)! Starten Sie danach die Aufgabe W.")
(0 "200705143303" "student" (document (concat "Das Kreuprodukt ist definiert als A" (menu-produkt) "B :=" (menu-menge "(a,b)" (concat "a " (menu-element) " A" (menu-und) "b" (menu-element) "B")))) ("SPELLOUT" "Kreuprodukt") "")
(0 "200705143303" "tutor" "Die Eingabe wird nicht ausgewertet (keine Aufgabe).")
(1 "200705143529")
(1 "200705143745" "student" (document "Wie fange ich an") ("SPELLOUT"))
(1 "200705143853" "tutor" (document (default-input "Eingabe" (document "Sie muessen die Gleichheit von zwei Mengen zeigen." "Bitte schauen Sie nach, wie man das macht.") "Korrektheit" "k.A." "Detail" "k.A." "Relevanz" "k.A.")) ("SPELLOUT"))
(1 "200705144126" "student" (document (concat "Es muss gelten, dass (a,b) " (menu-element) "(R" (menu-verknuepft) "S)" (menu-invers) " " (menu-und) "(a,b) " (menu-element) "S" (rsup "-1") "<circ>R" (rsup (concat "-1" (line-break))))) ("SPELLOUT"))
(1 "200705144327" "tutor" (document (default-input "Eingabe" (document (concat "Nicht ganz richtig. Sie muessen zeigen, dass wenn (a,b) " (menu-element) "(R" (menu-verknuepft) "S)" (menu-invers) " ") (concat "dann auch (a,b) " (menu-element) "S" (rsup "-1") "<circ>R" (rsup (concat "-1" (line-break))) " (und umgekehrt).")) "Korrektheit" "teilw. korrekt" "Detail" "angemessen" "Relevanz" "relevant")) ("SPELLOUT"))
(1 "200705144831" "student" (document (concat "(a,b) " (menu-element) " S" (menu-invers) ", heisst, dass (b,a)" (menu-element) "S. Das Gleiche gilt fuer R, also (b,a) " (menu-element) "R.")) ("SPELLOUT"))

(1 "200705145243" "tutor" (document (default-input "Eingabe" (document "Der erste Satz ist zwar korrekt, aber nicht direkt" "relevant." "Ich schlage folgenden Anfang vor:" (concat "Sei (a,b) " (menu-element) "(R" (menu-verknuepft) "S)" (menu-invers) " . ") "" "Fahren Sie jetzt fort. Schauen Sie " "sich die Definitionen der inversen Relation und des" "Relationsprodukts an.") "Korrektheit" "teilw. korrekt" "Detail" "angemessen" "Relevanz" "eingeschraenkt relevant")) ("SPELLOUT" "Relationsprodukts") "SPELLCHECKER-TO-nicht erkannte Worte: (Relationsprodukts)")
(1 "200705145326" "tutor" (document (default-input "Eingabe" (document "Der erste Satz ist zwar korrekt, aber nicht direkt" "relevant." "Ich schlage folgenden Anfang vor:" (concat "Sei (a,b) " (menu-element) "(R" (menu-verknuepft) "S)" (menu-invers) " . ") "" "Fahren Sie jetzt fort. Schauen Sie " "sich die Definitionen der inversen Relation und des" "Relationenprodukts an.") "Korrektheit" "teilw. korrekt" "Detail" "angemessen" "Relevanz" "eingeschraenkt relevant")) ("SPELLOUT"))
(1 "200705145343" "student" (document (concat "Es muss gelten, dass (a,b) " (menu-element) " S, (b,c) " (menu-element) " R, also ist (b,a) " (menu-element) " S" (menu-invers) " und (c,b) " (menu-element) " R" (menu-invers) " ")) ("SPELLOUT"))
(1 "200705145622" "tutor" (document (default-input "Eingabe" (document "Ja, wenn" (with "color" "magenta" (document (concat " (a,b) " (menu-element) " S und (b,c) " (menu-element) " R, dann ist (b,a) " (menu-element) " S" (menu-invers) " und (c,b) " (menu-element) " R" (menu-invers) " ")))) "Korrektheit" "korrekt" "Detail" "angemessen" "Relevanz" "relevant")) ("SPELLOUT"))
(1 "200705145659" "student" (document (concat "wenn (a,b) " (menu-element) " (R" (menu-verknuepft) "S)" (menu-invers) ", dann gilt, dass (b,a) " (menu-element) " (R" (menu-verknuepft) "S) (inverse Relation)")) ("SPELLOUT"))
(1 "200705145907" "tutor" (document (default-input "Eingabe" "Korrekt." "Korrektheit" "korrekt" "Detail" "angemessen" "Relevanz" "relevant")) ("SPELLOUT"))
(1 "200705145959" "student" (document (concat "Dann gilt auch, " (menu-existiert) "x, sodass (b,x) " (menu-element) " R und (x,a) " (menu-element) " S") (concat "Daraus folgt, dass (x,b) " (menu-element) " R" (menu-invers) " und (a,x) " (menu-element) " S" (menu-invers) " (inversion)")) ("SPELLOUT" "inversion") "")
(1 "200705150047" "tutor" (document (default-input "Eingabe" "Ja, das ist korrekt." "Korrektheit" "korrekt" "Detail" "angemessen" "Relevanz" "relevant")) ("SPELLOUT"))
(1 "200705150129" "student" (document (concat "Daraus ergibt sich, dass (a,b) " (menu-element) "S" (rsup "-1") "<circ>R" (rsup (concat "-1" (line-break))) ", q.e.d.")) ("SPELLOUT"))
(1 "200705150215" "tutor" (document (default-input "Eingabe" "Korrekt. Aufgabe geloest. Herzlichen Glueckwunsch!" "Korrektheit" "korrekt" "Detail" "angemessen" "Relevanz" "relevant")) ("SPELLOUT"))

(2 "200705150221")

(2 "200705150537" "student" (document (concat "(a,b) " (menu-element) "(R<cup>S), wenn (a,b) " (menu-element) "R oder (a,b) <in> S")) ("SPELLOUT"))

(2 "200705150615" "tutor" (document (default-input "Eingabe" "Korrekt." "Korrektheit" "korrekt" "Detail" "angemessen" "Relevanz" "relevant")) ("SPELLOUT"))

(2 "200705150953" "student" (document (concat (menu-existiert) "x, sodass (a,x) <in> (R<cup>S) und (x,b) <in> T")) ("SPELLOUT"))

(2 "200705151051" "tutor" (document (default-input "Eingabe" "Aus was folgt dies?" "Korrektheit" "korrekt" "Detail" "zu grobschrittig" "Relevanz" "relevant")) ("SPELLOUT"))

(2 "200705151306" "student" (document (concat "Das folgt aus der Definition des Relationsprodukts fuer binaere Relationen: (a,x) <in>R und (x,b)<in>S, dann R" (menu-verknuepft) "S")) ("SPELLOUT" "Relationsprodukts") "")

(2 "200705151511" "tutor" (document (default-input "Eingabe" (concat "Genauer: es folgt aus " (menu-paar "a" "b") (menu-element) "(R<cup>S)<circ>T") "Korrektheit" "korrekt" "Detail" "angemessen" "Relevanz" "eingeschraenkt relevant")) ("SPELLOUT"))

(2 "200705151639" "student" (document "AlsoKombination der Schritte: fuer (a,b) <in> (R<cup>S)<circ>T gilt:" (concat "entweder (a,x)<in> R oder (a,x)<in>S ,weil (a,b) " (menu-element) "(R<cup>S), wenn (a,b) " (menu-element) "R oder (a,b) <in> S") "und gleichzeitig gilt (x, b) <in>T") ("SPELLOUT" "AlsoKombination") "")

(2 "200705151842" "tutor" (document (default-input "Eingabe" (document "Diese Aussage ist nur teilweise korrekt. Bitte seien Sie " "genauer.") "Korrektheit" "teilw. korrekt" "Detail" "zu grobschrittig" "Relevanz" "relevant")) ("SPELLOUT"))

(2 "200705151853" "student" (document (concat "Das ist das gleiche wie (a,x) <in>R " (menu-und) "(x,b) <in> T oder:") (concat "(a,x) <in> S " (menu-und) " (x,b) <in> T")) ("SPELLOUT"))

(2 "200705151925" "tutor" (document (default-input "Eingabe" "Richtig." "Korrektheit" "korrekt" "Detail" "angemessen" "Relevanz" "relevant")) ("SPELLOUT"))

(2 "200705152026" "student" (document (concat "Daraus folgt, dass (a,b) <in> R" (menu-verknuepft) "T oder (a,b) <in> S" (menu-verknuepft) "T")) ("SPELLOUT"))

(2 "200705152059" "tutor" (document (default-input "Eingabe" "Richtig! Sie sind fast fertig." "Korrektheit" "korrekt" "Detail" "angemessen" "Relevanz" "relevant")) ("SPELLOUT"))

(2 "200705152132" "student" (document (concat "Also (a,b) <in> R" (menu-verknuepft) "T " (menu-schnitt) " S" (menu-verknuepft) "T q.e.d.")) ("SPELLOUT"))

(2 "200705152219" "tutor" (document (default-input "Eingabe" "Das ist nicht ganz korrekt. " "Korrektheit" "inkorrekt" "Detail" "angemessen" "Relevanz" "relevant")) ("SPELLOUT"))

(2 "200705152223" "student" (document (concat "nein, ich meine (a,b) <in> R" (menu-verknuepft) "T " (menu-vereinigung) " S" (menu-verknuepft) "T q.e.d.")) ("SPELLOUT"))
(2 "200705152307" "tutor" (document (default-input "Eingabe" "Das ist korrekt. Aufgabe geloest. Herzlichen Glueckwunsch!" "Korrektheit" "korrekt" "Detail" "angemessen" "Relevanz" "relevant")) ("SPELLOUT"))
(3 "200705152310")
(3 "200705152512" "student" (document "Fuer (R<cup>S)<circ>T gilt : " (concat "(a,x) <in>R " (menu-und) "(x,b) <in> T") "oder:" (concat "(a,x) <in> S " (menu-und) " (x,b) <in> T")) ("SPELLOUT"))
(3 "200705152549" "tutor" (document (default-input "Eingabe" "Das ist richtig." "Korrektheit" "korrekt" "Detail" "angemessen" "Relevanz" "relevant")) ("SPELL-OUT"))
(3 "200705152553" "student" (document "Darf ich vorherige Beweise benutzen?") ("SPELLOUT"))
(3 "200705152613" "tutor" (document (default-input "Eingabe" "Ja, das duerfen Sie." "Korrektheit" "k.A." "Detail" "k.A." "Relevanz" "relevant")) ("SPELLOUT"))
(3 "200705152958" "student" (document (concat "Hm, naja, fuer (T" (rsup "-1") "<circ>S" (rsup "-1") ")" (rsup "-1") " gilt : wenn (a,b) <in> (T" (rsup "-1") "<circ>S" (rsup "-1") ")" (rsup "-1") ",") (concat "dann (b,a)<in> (T" (rsup "-1") "<circ>S" (rsup "-1") ")")) ("SPELLOUT" "Hm" "naja") "")
(3 "200705153030" "tutor" (document (default-input "Eingabe" "Korrekt." "Korrektheit" "korrekt" "Detail" "angemessen" "Relevanz" "relevant")) ("SPELLOUT"))
(3 "200705153129" "student" (document (concat "Also (b,a) <in> (T" (rsup "-1") "<circ>S" (rsup "-1") ") oder (b,a) <in> (T" (rsup "-1") "<circ>R" (rsup "-1") ") (wegen Vereinigungsregel)")) ("SPELLOUT" "Vereinigungsregel") "")
(3 "200705153222" "tutor" (document (default-input "Eingabe" "Korrekt." "Korrektheit" "korrekt" "Detail" "angemessen" "Relevanz" "relevant")) ("SPELLOUT"))
(3 "200705153539" "student" (document (concat "Der Beweis in Aufgabe W zeigt, dass (T" (rsup "-1") "<circ>S" (rsup "-1") ") = (T" (menu-verknuepft) "S)" (menu-invers) ", ") (concat "also (b,a) <in> (T" (menu-verknuepft) "S)" (menu-invers) " oder (b,a) <in> (T" (menu-verknuepft) "R)" (menu-invers))) ("SPELLOUT"))
(3 "200705153635" "tutor" (document (default-input "Eingabe" "Ja, das ist richtig." "Korrektheit" "korrekt" "Detail" "angemessen" "Relevanz" "relevant")) ("SPELL-OUT"))
(3 "200705153649" "student" (document (concat "Daraus folgt, dass (a,b) <in> (T" (menu-verknuepft) "S) oder (a,b) <in> (T" (menu-verknuepft) "R) (inversion)")) ("SPELLOUT" "inversion") "")
(3 "200705153859" "tutor" (document (default-input "Eingabe" (document "Sorry. Der Schluss in 20 war nicht korrekt. Bitte gehen" "Sie zurueck zu 20.") "Korrektheit"

"k.A." "Detail" "k.A." "Relevanz" "k.A.")) ("SPELLOUT" "Sorry") "SPELLCHECKER-TO-nicht erkannte Worte: (Sorry)")

(3 "200705153958" "tutor" (document (default-input "Eingabe" (document "Entschuldigen Sie. Der Schluss in 20 war nicht korrekt. Bitte gehen" "Sie zurueck zu 20.") "Korrektheit" "k.A." "Detail" "k.A." "Relevanz" "k.A.")) ("SPELLOUT"))

(3 "200705154037" "student" (document (concat "Also (a,b) <in> T" (menu-verknuepft) "S " (menu-vereinigung) "T" (menu-verknuepft) "R") "und somit auch <in>(R<cup>S)<circ>T, wie der Beweis der Aufgabe 1 zeigte") ("SPELLOUT"))

(3 "200705154117" "tutor" (document (default-input "Eingabe" "Nicht richtig. Bitte zurueck zu 20." "Korrektheit" "inkorrekt" "Detail" "angemessen" "Relevanz" "relevant")) ("SPELLOUT"))

(3 "200705154132" "student" (document "[Student will Programm beenden]") ("SPELLOUT"))

(5 "200705154138" "server" "Session beendet")

Appendix D

Inference Rules in PSYCOP

Source: Lance J. Rips. The psychology of proof: deductive reasoning in human thinking. MIT Press, Cambridge, MA, 1994, pages 113f, 116ff, 129, 191f, 195, 197-200, 202.

D.1 PSYCOP's Forward Routines

Forward IF Elimination

$$\frac{\text{IF P THEN Q} \quad \text{P}}{\text{Q}}$$

(a) If a sentence of the form IF P THEN Q holds in some domain D,
(b) and P holds in D,
(c) and Q does not yet hold in D,
(d) then add Q to D.

Forward AND Elimination

$$\frac{\text{P AND Q}}{\text{P}} \qquad \frac{\text{P AND Q}}{\text{Q}}$$

(a) If a sentence of the form P AND Q holds in some domain D,
(b) then if P does not yet hold in D,
(c) then add P to D,
(d) and if Q does not yet hold in D,
(e) then add Q to D.

Forward Double Negation Elimination

$$\frac{\text{NOT NOT P}}{\text{P}}$$

(a) If a sentence of the form NOT NOT P holds in some domain D,

(b) and P does not yet hold in D,

(c) then add P to D.

Forward Disjunctive Syllogism

$$\frac{\begin{array}{l}\text{P OR Q}\\ \text{NOT Q}\end{array}}{\text{P}}$$

$$\frac{\begin{array}{l}\text{P OR Q}\\ \text{NOT P}\end{array}}{\text{Q}}$$

(a) If a sentence of the form P OR Q holds in some domain D,

(b) then if NOT P holds in D and Q does not yet hold in D,

(c) then add Q to D.

(d) Else, if NOT Q holds in D and and P does not yet hold in D,

(e) then add P to D.

Forward Disjunctive Modus Ponens

$$\frac{\begin{array}{l}\text{IF P OR Q THEN R}\\ \text{P}\end{array}}{\text{R}}$$

$$\frac{\begin{array}{l}\text{IF P OR Q THEN R}\\ \text{Q}\end{array}}{\text{R}}$$

(a) If a sentence of the form IF P OR Q THEN R holds in some domain D,

(b) and P or Q also holds in D,

(c) and R does not yet hold in D,

(d) then add R to D.

Forward Conjunctive Modus Ponens

$$\frac{\begin{array}{l}\text{IF P AND Q THEN R}\\ \text{P}\\ \text{Q}\end{array}}{\text{R}}$$

(a) If a sentence of the form IF P AND Q THEN R holds in some domain D,

(b) and P holds in D,

(c) and Q holds in D,

(d) and R does not yet hold in D,

(e) then add R to D.

Forward DeMorgan (NOT over AND)

$$\begin{array}{c} \text{NOT (P AND Q)} \\ \hline \text{(NOT P) OR (NOT Q)} \end{array}$$

(a) If a sentence of the form NOT (P AND Q) holds in some domain D,
(b) and (NOT P) OR (NOT Q) does not yet hold in D,
(c) then add (NOT P) OR (NOT Q) to D.

Forward DeMorgan (NOT over OR)

$$\begin{array}{c} \text{NOT (P OR Q)} \\ \hline \text{NOT P} \end{array}$$

$$\begin{array}{c} \text{NOT (P OR Q)} \\ \hline \text{NOT Q} \end{array}$$

(a) If a sentence of the form NOT (P OR Q) holds in some domain D,
(b) then if NOT P does not yet hold in D
(c) then add NOT P to D.
(d) and if NOT Q does not yet hold in D,
(e) then add NOT Q to D.

Forward Conjunctive Syllogism

$$\begin{array}{l} \text{NOT (P AND Q)} \\ \text{P} \\ \hline \qquad \text{NOT Q} \end{array}$$

$$\begin{array}{l} \text{NOT (P AND Q)} \\ \text{Q} \\ \hline \qquad \text{NOT P} \end{array}$$

(a) If a sentence of the form NOT (P AND Q) holds in some domain D,
(b) then if P also holds in D
(c) and NOT Q does not yet hold in D,
(d) then add NOT Q to D.
(e) Else, if Q holds in D,
(f) and NOT P does not yet hold in D,
(g) then add NOT P to D.

Forward Dilemma

$$\frac{\begin{array}{l}\text{P OR Q}\\ \text{IF P THEN R}\\ \text{IF Q THEN R}\end{array}}{\text{R}}$$

(a) If a sentence of the form P OR Q holds in some domain D,

(b) and IF P THEN R holds in D,

(c) and IF Q THEN R holds in D,

(d) and R does not yet hold in D,

(e) then add R to D.

D.2 PSYCOP's Backward Rules

Backward IF Introduction (Conditionalization)

$$\frac{\begin{array}{c}+\text{P}\\ \vdots\\ \text{Q}\end{array}}{\text{IF P THEN Q}}$$

(a) Set D to domain of current goal.

(b) If current goal is of the form IF P THEN Q,

(c) and neither D nor its superdomains nor its immediate subdomains contain supposition P and subgoal Q,

(d) and IF P THEN Q is a subformula of the premises or conclusion,

(e) then set up a subdomain of D, D', with supposition P.

(f) Add the subgoal of proving Q in D' to the list of subgoals.

Backward NOT Elimination

$$\frac{\begin{array}{c} +\text{NOT P} \\ \vdots \\ \text{Q AND (NOT Q)} \end{array}}{\text{P}}$$

(a) Set P to current goal and D to its domain.

(b) If P is a subformula of the premises or conclusion,

(c) and Q is an atomic subformula in the premises or conclusion,

(d) and neither D nor its superdomains nor its immediate subdomains contain supposition NOT P and subgoal Q AND (NOT Q),

(e) then set up a subdomain of D, D', with supposition NOT P,

(f) and add the subgoal of proving Q AND (NOT Q) in D' to the list of subgoals.

Backward NOT Introduction

$$\frac{\begin{array}{c} +\text{P} \\ \vdots \\ \text{Q AND (NOT Q)} \end{array}}{\text{NOT P}}$$

(a) Set D to domain of current goal.

(b) If current goal is of the form NOT P,

(c) and P is a subformula of the premises or conclusion,

(d) and Q is an atomic subformula of the premises or conclusion,

(e) and neither D nor its superdomains nor its immediate subdomains contain supposition P and subgoal Q AND (NOT Q),

(f) then set up a subdomain of D, D', with supposition P,

(g) and add the subgoal of proving Q AND (NOT Q) in D' to the list of subgoals.

Backward OR Elimination

$$
\begin{array}{l}
\text{P OR Q} \\
+P \\
\vdots \\
R \\
+Q \\
\vdots \\
\underline{R \qquad\qquad\qquad} \\
\qquad\quad \text{R}
\end{array}
$$

(a) Set D to domain of current goal.

(b) Set R to current goal.

(c) If a sentence of the form P OR Q holds in D,

(d) and both P and Q are subformulas or negations of subformulas of the premises or conclusion,

(e) and R is a subformula or negation of a subformula of the premises or conclusion,

(f) and neither D nor its superdomains nor its immediate subdomains contain supposition P and subgoal R,

(g) and neither D nor its superdomains nor its immediate subdomains contain supposition Q and subgoal R,

(h) then set up a subdomain of D, D', with supposition P,

(i) and add the subgoal of proving R in D' to the list of subgoals.

(j) If the subgoal in (i) succeeds,

(k) then set up another subdomain of D, D", with supposition Q,

(l) and add the subgoal of proving R in D" to the list of subgoals.

Backward AND Introduction

$$\frac{\begin{array}{l}\mathrm{P}\\ \mathrm{Q}\end{array}}{\mathrm{P\ AND\ Q}}$$

(a) Set D to domain of current goal.
(b) If current goal is of the form P AND Q,
(c) and D does not yet contain the subgoal P,
(d) then add the subgoal of proving P in D to the list of subgoals.
(e) If the subgoal in (d) succeeds,
(f) and D does not yet contain the subgoal Q,
(g) then add the subgoal of proving Q in D to the list of subgoals.

Backward OR Introduction

$$\frac{\mathrm{P}}{\mathrm{P\ OR\ Q}}$$

$$\frac{\mathrm{Q}}{\mathrm{P\ OR\ Q}}$$

(a) Set D to domain of current goal.
(b) If current goal is of the form P OR Q,
(c) and D does not yet contain the subgoal P,
(d) then add the subgoal of proving P in D to the list of subgoals.
(e) If the subgoal in (d) fails,
(f) and D does not yet contain the subgoal Q,
(g) then add the subgoal of proving Q in D to the list of subgoals.

Backward IF Elimination (modus ponens)

$$\frac{\begin{array}{l}\mathrm{IF\ P\ THEN\ Q}\\ \mathrm{P}\end{array}}{\mathrm{Q}}$$

(a) Set D to domain of current goal.
(b) Set Q to current goal.
(c) If the sentence IF P THEN Q holds in D,
(d) and D does not yet contain the subgoal P,
(e) then add P to the list of subgoals.

Backward AND Elimination

$$\frac{\text{P AND Q}}{\text{P}}$$

$$\frac{\text{Q AND P}}{\text{P}}$$

(a) Set D to domain of current goal.
(b) Set P to current goal.
(c) If the sentence P AND Q is a subformula of a sentence that holds in D,
(d) and D does not yet contain the subgoal P AND Q,
(e) then add P AND Q to the list of subgoals.
(f) If the subgoal in (e) fails,
(g) and the sentence Q AND P is a subformula of a sentence that holds in D,
(h) and D does not yet contain the subgoal Q AND P,
(i) then add Q AND P to the list of subgoals.

Backward Double Negation Elimination

$$\frac{\text{NOT NOT P}}{\text{P}}$$

(a) Set D to domain of current goal.
(b) Set P to the current goal.
(c) If the sentence NOT NOT P is a subformula of a sentence that holds in D,
(d) and D does not yet contain the subgoal NOT NOT P,
(e) then add NOT NOT P to the list of subgoals.

Backward Disjunctive Modus Ponens

$$\frac{\text{IF P OR Q THEN R} \quad \text{P}}{\text{R}}$$

$$\frac{\text{IF P OR Q THEN R} \quad \text{Q}}{\text{R}}$$

(a) Set D to domain of current goal.
(b) Set R to current goal.
(c) If the sentence IF P OR Q THEN R holds in D,
(d) and D does not yet contain the subgoal P,
(e) then add P to the list of subgoals.
(f) If the subgoal in (e) fails,
(g) and D does not yet contain the subgoal Q,
(h) then add Q to the list of subgoals.

Backward Disjunctive Syllogism

$$\frac{\text{P OR Q} \quad \text{NOT P}}{\text{Q}}$$

$$\frac{\text{Q OR P} \quad \text{NOT P}}{\text{Q}}$$

(a) Set D to domain of current goal.
(b) Set Q to to current goal.
(c) If the sentence of the form P OR Q or Q OR P holds in D,
(d) and NOT P is a subformula of a sentence that holds in D,
(e) and D does not yet contain the subgoal NOT P,
(f) then add NOT P to the list of subgoals.

Backward Conjunctive Syllogism

$$\frac{\text{NOT (P AND Q)} \quad \text{P}}{\text{NOT Q}}$$

$$\frac{\text{NOT (Q AND P)} \quad \text{P}}{\text{NOT Q}}$$

(a) Set D to domain of current goal.
(b) If the current goal is of the form NOT Q
(c) and either of the sentences NOT (P AND Q) or NOT (Q AND P) holds in D,
(d) and D does not yet contain the subgoal P,
(e) then add P to the list of subgoals.

Backward DeMorgan (NOT over AND)

$$\frac{\text{NOT (P AND Q)}}{\text{(NOT P) OR (NOT Q)}}$$

(a) Set D to domain of current goal.
(b) If current goal is of the form (NOT P) OR (NOT Q),
(c) and NOT (P AND Q) is a subformula of a sentence that holds in D,
(d) then add NOT (P AND Q) to the list of subgoals.

Backward DeMorgan (NOT over OR)

$$\frac{\text{NOT (P OR Q)}}{\text{(NOT P) AND (NOT Q)}}$$

(a) Set D to domain of current goal.
(b) If current goal is of the form (NOT P) AND (NOT Q),
(c) and NOT (P OR Q) is a subformula of a sentence that holds in D,
(d) then add NOT (P OR Q) to the list of subgoals.

D.3 PSYCOP+

Negated Conditional Transformation

$$\frac{\text{NOT (IF P THEN Q)}}{\text{P}}$$

$$\frac{\text{NOT (IF P THEN Q)}}{\text{NOT Q}}$$

(a) If a sentence of the form NOT (IF P THEN Q) holds in some domain D,
(b) then if P does not yet hold in D,
(c) then add P to D,
(d) and if NOT Q does not yet hold in D,
(e) then add NOT Q to D.

Conditional Transformation

$$\frac{\text{IF P THEN Q}}{\text{(NOT P) OR Q}}$$

(a) If a sentence of the form IF P THEN Q holds in some domain D,

(b) and if (NOT P) OR Q does not yet hold in D,

(c) then add (NOT P) OR Q to D.

D.4 Matching Rules

Matching rules for predicate logic. In the rules, P(n) represents an arbitrary quantifier-free sentence that contains argument n. P and P' are said to be isomorphic if they are identical except for the particular variables and temporary names they contain (i.e., they must have the same predicates and connectives in the same positions). Some backward rules (e.g., IF Elimination) require a subgoal to match a subformula of an assertion, and in that case the phrase "target formula" in the following rules refers to that subformula. Otherwise, the target formula is a complete assertion.

Matching 1 (variables in subgoal to variables in assertion)

Conditions:

(a) P(X) is a target formula and P'(y) is an isomorphic subgoal.
(b) x and y are variables.
(c) wherever x appears in (a nonsubscript position in) P(x) either y or a temporary name t_y with y as a subscript appears in P'(y).
(d) if t_y appears in the same position as x, then t_y only appears in positions occupied by variables.
(e) wherever x appears as a subscript in P(x), y appears as a subscript in P'(y).

Action:

Add the subgoal of proving P'(x) to the list of subgoals, where P'(x) is the result of:

(a) substituting x for all occurences of y in P'(y)
(b) substituting x for all occurences of t_y in P'(y) if t_y appeared in some of the same positions as x.
(c) substituting x for y in P'(y) at each remaining subscript position that y occupies in P'(y).

Matching 2 (temporary names in subgoal to temporary or permanent names in assertion)

Conditions:
- (a) P(n) is a target formula and P'(t) is an isomorphic subgoal.
- (b) t is a temporary name, and n is a (temporary or permanent) name.
- (c) any subscripts of n are included among the subscripts of t.
- (d) t is not of the form $\hat{a}$.
- (e) either n or a variable appears in P(n) in all positions that t occupies in P'(t).

Action: Add the subgoal of proving P'(n) to the list of subgoals, where P'(n) is the result of substituting n for t in P'(t) at all positions that t occupies in P'(t).

Matching 3 (permanent names in subgoal to variables in assertion)

Conditions:
- (a) P(x) is a target formula and P'(m) is an isomorphic subgoal.
- (b) x is a variable and m is a permanent name.
- (c) m appears in P'(m) in each nonsubscript position that x occupies in P(x).
- (d) if a temporary name t appears in P'(m) in the same position that an x-subscripted temporary name t_x appears in P(x), then any additional tokens of t also appear in positions occupied by t_x.

Action: Add the subgoal of proving P'(x) to the list of subgoals, where P'(x) is the result of:
- (a) substituting x for m in P'(m) at each nonsubscript position that x occupies in P(x).
- (b) substituting t' for t in P'(m) wherever t occurs in the same position as an x-subscripted temporary name in P(x) and where t' is a new temporary name with subscripts consisting of x and all subscripts of t.

Matching 4 (temporary names in subgoal to variables in assertion)

Conditions:

(a) $P(x_1, ..., x_k)$ is a target formula and $P'(t)$ is an isomorphic subgoal.

(b) $x_1, ..., x_k$ are variables and t is a temporary name.

(c) x_1 or x_2 or $...x_k$ appears in non-subscript positions in $P(x_1, ..., x_k)$ at each position that t occupies in $P'(t)$.

(d) if a temporary name t' appears in $P'(t)$ in the same position that an x_i-subscripted temporary name t_{x_i} appears in $P(x_1, ..., x_k)$, then any additional tokens of t' also appear in positions occupied by t_{x_i}.

Action:

Add the subgoal of proving $P'(x_1, ..., x_k)$ to the list of subgoals, where $P'(x_1, ..., x_k)$ is the result of:

(a) substituting x_i for t wherever t occupies the same position in $P'(t)$ than x_i occupies in $P(x_1, ..., x_k)$.

(b) substituting t'' for t' in $P'(t)$ wherever t' occurs in the same position as an x_i-subscripted temporary name in $P(x_1, ..., x_k)$ and where t'' is a new temporary name whose subscripts consist of x_i and all subscripts of t'.

D.5 New Forward Inference Rules for Predicate Logic

Transitivity

IF F(x) THEN G(x)
IF G(y) THEN H(y)
___
IF F(z) THEN H(z)

(a) If sentences of the form IF F(x) THEN G(x) and IF G(y) THEN H(y) hold in some domain D,

(b) and IF F(z) THEN H(z) does not yet hold in D,

(c) then add IF F(z) THEN H(z) to D.

Exclusivity

$$\frac{\text{IF F(x) THEN G(x)} \quad \text{NOT (G(y) AND H(y))}}{\text{NOT (F(z) AND H(z))}}$$

(a) If sentences of the form IF F(x) THEN G(x) and NOT (G(y) AND H(y)) hold in some domain D,
(b) and NOT (F(z) AND H(z)) does not yet hold in D,
(c) then add NOT(F(z) AND H(z)) to D.

Conversion

$$\frac{\text{NOT (F(x) AND G(x))}}{\text{NOT (G(y) AND F(y))}}$$

(a) If sentences of the form NOT (F(x) AND G(x)) holds in some domain D,
(b) and NOT (G(y) AND F(y)) does not yet hold in D,
(c) then add NOT (G(y) AND F(y)) to D.

D.6 Backward Inference Rules for Predicate Logic

Backward inference rules for predicate logic. In these an asterisk indicates the result of inversing arguments by means of the procedure described in the text. P and P' denote isomorphic sentences (ones which are identical except for their variables and names). P* is the result of applying the Argument Reversal procedure to P, unless otherwise indicated. Notational variants are isomorphic sentences that differ only by substitution of variables for other variables or temporary names for other temporary names. Conditions for matching isomorphic sentences are described in Section D.4. Procedures for Argument Reversal and Subscript Adjustment appear in Section D.7.

Backward IF Elimination (Modus Ponens)

(a) Set R to the current goal and set D to its domain.
(b) If R can be matched to R' for some sentence IF P' THEN R' that holds in D,
(c) then go to Step (e).

(d) Else, return failure.

(e) If P' and R' share variables and one or more names or variables in R matched these variables,

(f) then set P to the result of substituting those names or variables for the corresponding variables in P'. Label the substituting arguments the matched arguments of P and the residual arguments the unmatched arguments of P.

(g) Else, set P to P'. Label all its arguments as unmatched.

(h) Apply Argument Reversal to unmatched arguments of P.

(i) Apply Subscript Adjustment to output of Step (h). Call the result P*.

(j) If D does not yet contain the subgoal P* or a notational variant,

(k) then add the subgoal of proving P* in D to the list of subgoals.

Backward IF Introduction (Conditionalization)

(a) Set D to domain of current goal.

(b) If current goal is of the form IF P THEN R,

(c) and neither D nor its superdomains nor its immediate subdomains contains both supposition P and subgoal R (or notational variants),

(d) and IF P THEN R is a subformula of the premises or conclusion,

(e) then let P' be the result of substituting in P new tented temporary names for any variables that P shares with R,

(f) and label substituting arguments matched arguments and the residual arguments unmatched,

(g) and set up a subdomain of D, D', with supposition P* (where P* is the result of applying the Argument Reversal procedure to any unmatched arguments in P').

(h) Add the subgoal of proving R in D' to the list of subgoals.

Backward NOT Elimination

(a) Set P to current goal and D to its domain.

(b) If P is a subformula of the premises or conclusion,

(c) and Q is an atomic subformula in the premises or conclusion,

(d) and neither D nor its superdomains nor its immediate subdomains contains both supposition NOT P* and subgoal Q or supposition NOT P* and subgoal NOT Q* (or notational variants),

(e) then set up a subdomain of D, D', with supposition NOT P*,

(f) and add the subgoal of proving Q AND NOT Q* in D' to the list of subgoals.

(g) If the subgoals in (f) fails,

(h) and neither D nor its superdomains nor its immediate subdomains contains both supposition NOT P* and subgoal Q* or supposition NOT P* and subgoal NOT Q (or notational variants),

(i) then set up a subdomain of D, D", with supposition NOT P*,

(j) and add the subgoal of proving Q* AND NOT Q in D" to the list of subgoals.

Backward NOT Introduction

(a) Set D to domain of current goal.

(b) If current goal is of the form NOT P,

(c) and P is a subformula (or notational variant of a subformula) of the premises or conclusion,

(d) and Q is an atomic subformula of the premises or conclusion,

(e) and neither D nor its superdomains nor its immediate subdomains contains both supposition P* and and subgoal Q, or supposition P* and subgoal NOT Q* (or notational variants),

(f) then set up a subdomain of D, D', with supposition P*,

(g) and add the subgoal of proving Q AND NOT Q* in D' to the list of subgoals.

(h) If the subgoal in (g) fails,

(i) and neither D nor its superdomains nor its immediate subdomains contains both supposition P* and and subgoal Q* or supposition P* and subgoal NOT Q (or notational variants),

(j) then set up a subdomain of D, D", with supposition P*,

(k) and add the subgoal of proving Q* AND NOT Q in D" to the list of subgoals.

Backward AND Introduction

(a) Set D to domain of current goal.
(b) If current goal is of the form P AND Q,
(c) and D does not yet contain the subgoal P,
(d) then add the subgoal of proving P in D to the list of subgoals.
(e) If the goal in (d) succeeds,
(f) and P is matched to P',
and P and Q share temporary names,
(g) then set Q' to the result of substituting in Q any names that were matched to those temporary names.
(h) Else, set Q' to Q.
(i) If D does not yet contain the subgoal Q',
(j) then add the subgoal of proving Q' in D to the list of subgoals.

Backward OR Elimination

(a) Set R to current goal and set D to its domain.
(b) If a sentence of the form P OR Q holds in D,
(c) and both P and Q are subformulas or negations of subformulas of the premises or conclusion (or notational variants of a subformula),
(d) and neither D nor its superdomains nor its immediate subdomains contains both suppositions P and subformula R (or notational variants)
(e) and neither D nor its superdomains nor its immediate subdomains contains both suppositions Q and subformula R (or notational variants)
(f) then replace any variables that appear in both P and Q with temporary names that do not yet appear in the proof. Call the result P' OR Q'.
(g) Set up a subdomain of D, D', with supposition P'.
(h) and add the subgoal of proving R in D' to the list of subgoals.
(i) If the subgoal in (h) succeeds,
(j) then set up a subdomain of D, D'', with supposition Q',
(k) and add the subgoal of proving R in D'' to the list of subgoals.

Backward OR Introduction

(a) Set D to domain of current goal.
(b) If current goal is of the form P OR Q,
(c) then if D does not yet contain the subgoal P or a notational variant,
(d) then add the subgoal of proving P in D to the list of subgoals.
(e) If the subgoal in (d) fails,
(f) and D does not yet contain the subgoal Q or a notational variant,
(g) then add the subgoal of proving Q in D to the list of subgoals.

Backward AND Elimination

(a) Set D to domain of current goal.
(b) Set P to current goal.
(c) If P can be matched to P' in some subformula P' AND Q' of a sentence that holds in D,
(d) and D does not yet contain the subgoal P' AND Q' or a notational variant,
(e) then add P' AND Q' to the list of subgoals.
(f) Else, if P can be matched to P' in some subformula Q' and P' of a sentence that holds in D,
(g) and D does not yet contain the subgoal Q' AND P' or a notational variant,
(h) then add Q' AND P' to the list of subgoals.

Backward Double Negation Elimination

(a) Set D to domain of current goal.
(b) Set P to current goal.
(c) If P matches P' in a subformula NOT NOT P' of a sentence that holds in D,
(d) and D does not yet contain the subgoal NOT NOT P' or a notational variant,
(e) then add NOT NOT P' to the list of subgoals.

Backward Disjunctive Syllogism

(a) Set D to domain of current goal.
(b) If the current goal Q matches Q' in a sentence (P' OR Q') or (Q' OR P') that holds in D,
(c) and NOT P' is isomorphic to a subformula of a sentence that holds in D,
(d) then go to Step (f).
(e) Else, return failure.
(f) If P' shares variables with Q' and one or more names or variables in Q matched these variables,
(g) then set P to the result of substituting those names or variables for the corresponding variables in P'. Label the substituting arguments, the matched arguments of P, and the residual arguments the unmatched arguments of P.
(h) Else, set P to P', labeling all its arguments as unmatched.
(i) Apply Argument Reversal to unmatched arguments of P and then Subscript Adjustment. Call the result P*.
(j) If D does not yet contain the subgoal NOT(P*) or a notational variant,
(k) then add the subgoal of proving NOT(P*) in D to the list of subgoals.

Backward Disjunctive Modus Ponens

(a) Set D to domain of current goal.
(b) Set R to current goal.
(c) If R can be matched to R' for some sentence IF P' OR Q' THEN R' that holds in D,
(d) then go to step (f).
(e) Else, return failure.
(f) If P' shares variables with R' and one or more names or variables in R matched these variables,
(g) then set P to the result of substituting those names or variables for the corresponding variables in P'. Label the substituting arguments, the matched arguments of P, and the residual arguments the unmatched arguments of P.
(h) Else, set P to P', labeling all its arguments as unmatched.
(i) Apply Argument Reversal and then Subscript Adjustment to P. Label the result P*.
(j) If D does not yet contain the subgoal P* or a notational variant,

(k) then add the subgoal of proving P* in D to the list of subgoals.

(l) If the subgoal in (k) fails,

(m) and Q' shares variables with R' and one or more names or variables in R matched these variables,

(n) then set Q to the result of substituting those names or variables for the corresponding variables in Q', labeling the arguments as in step (g).

(o) Else, set Q to Q', labeling all its arguments as unmatched.

(p) Apply Argument Reversal and then Subscript Adjustment to Q. Label the result Q*.

(q) If D does not yet contain the subgoal Q* or a notational variant,

(r) then add the subgoal of proving Q* in D to the list of subgoals.

Backward Conjunctive Syllogism

(a) Set D to domain of current goal.

(b) If the current goal is of the form NOT Q,

(c) and Q matches Q' in a sentence NOT (P' AND Q') or NOT (Q' AND P') that holds in D,

(d) then go to Step (f).

(e) Else, return failure.

(f) If P' shares variables with Q' and one or more variables in Q matched these variables,

(g) then set P to the result of substituting those names or variables for the corresponding variables in P'. Label the substituting arguments, the matched arguments of P, and the residual arguments the unmatched arguments of P.

(h) Else, set P to P', labeling all its arguments as unmatched.

(i) Apply Argument Reversal to unmatched arguments of P and then Subscript Adjustment. Call the result P*.

(j) If D does not yet contain the subgoal P* or a notational variant,

(k) then add the subgoal of proving P* in D to the list of subgoals.

Backward DeMorgan (NOT over AND)

(a) Set D to domain of current goal.

(b) If current goal is of the form (NOT P) OR (NOT Q),

(c) and some subformula of a sentence that holds in D is of the form NOT (P' AND Q'),

(d) and P AND Q matches P' AND Q',
(e) and D does not yet contain the subgoal NOT (P' AND Q') or a notational variant,
(f) then add NOT (P' AND Q') to the list of subgoals.

Backward DeMorgan (NOT over OR)

(a) Set D to domain of current goal.
(b) If current goal is of the form (NOT P) AND (NOT Q),
(c) and some subformula of a sentence that holds in D is of the form NOT (P' OR Q'),
(d) and P OR Q matches P' OR Q',
(e) and D does not yet contain the subgoal NOT (P' AND Q') or a notational variant,
(f) then add NOT (P' OR Q') to the list of subgoals.

D.7 Argument Reversal and Subscript Adjustment for Backward Rules

Argument Reversal

Let P be an arbitrary sentence in quantifier-free form. Then the following steps produce the argument-reversed, isomorphic sentence P_r:

1 Assign to each temporary name a_i that appears in P a variable y_i that has not yet appeared in the proof.

2 Assign to each variable x_i in P a temporary name b_i that has not yet appeared in the proof.

3 Let $\{a_i\}$ be the set of all temporary names in P that do not have x_i as a subscript. Let $\{y_i\}$ be the set of variables assigned in step 1 to the a_i's in $\{a_i\}$. Subscript b_i with the variables in $\{y_i\}$.

4 Replace each a_i in P with the y_i assigned to it in step 1. Replace each x_i in P with the b_i assigned to it in step 2, together with the subscripts computed in step 3. The result is P_r.

Subscript Adjustment

Let P be a sentence in quantifier-free form, and let P_r be the result of applying the Argument Reversal procedure above to unmatched arguments of P (i.e., arguments that do not arise from substitution). The following operations yield the subscript-adjusted sentence P_s when applied to P_r:

1 Let c_j be a temporary name in P_r that derives from reversing an unmatched variable x_j in P. For each such c_j, add to its subscripts any variable y_i in P_r, provided either of the following conditions is met:

   (a) y_i was matched to a variable x_i in P or y_i is a subscript of a temporary name that matched x_i in P, and no temporary name in P contains x_j but not x_i.

   (b) y_i is a subscript of a temporary name that matched some temporary name b_i in P, and b_i does not have x_j as a subscript.

2 Let d_j be a temporary name in P_r that derives from matching to some variable x_j of P. Then for each such d_j, add to its subscripts any variable x_i in P_r, provided all the following conditions are met:

(a) x_i derives from reversing a temporary name b_i from P.

(b) b_i does not have x_j as subscript.

(c) b_i does not have x_k as subscript, where y_k matched x_k and y_k is not a subscript of d_j.

Appendix E

Granularity Evaluation Data

Table E.1 shows the complexity scores for 20 analyzed proof steps w.r.t.

1 The number of PSYCOP proof steps that justify the original step when all definitions are removed [PSYCOP(1)].

2 The number of PSYCOP proof steps that justify the original step when as little definitions as possible are removed [PSYCOP(2)].

3 The number of Gentzen's natural deduction(ND) proof steps that justify the original step when all definitions are removed [ND(1)].

4 The number of Gentzen's ND proof steps that justify the original step when as little definitions as possible are removed [ND(2)].

5 The granularity rating assigned by the tutor during the experiment (D: too detailed, A: appropriate, C: too coarse-grained).

	Proof Step Count				Tutor's granularity rating
	PSYCOP(1)	PSYCOP(2)	Gentzen's ND(1)	Gentzen's ND(2)	(D: too detailed, A: appropriate, C: too coarse)
1	1	1	0	0	D
2	1	1	0	0	D
3	5	5	5	5	A
4	5	5	5	5	A
5	1	1	0	0	A
6	1	1	0	0	A
7	1	5	0	3	A
8	1	1	0	0	A
9	5	5	3	3	A
10	10	10	9	9	A
11	2	2	3	3	A
12	5	5	5	5	A
13	7	7	9	9	A
14	21	21	21	21	A
15	5	5	5	5	A
16	5	5	5	5	A
17	5	5	5	5	A
18	21	21	21	21	C
19	9	9	7	7	C
20	5	5	3	3	C

Figure E.1: Granularity Evaluation Data