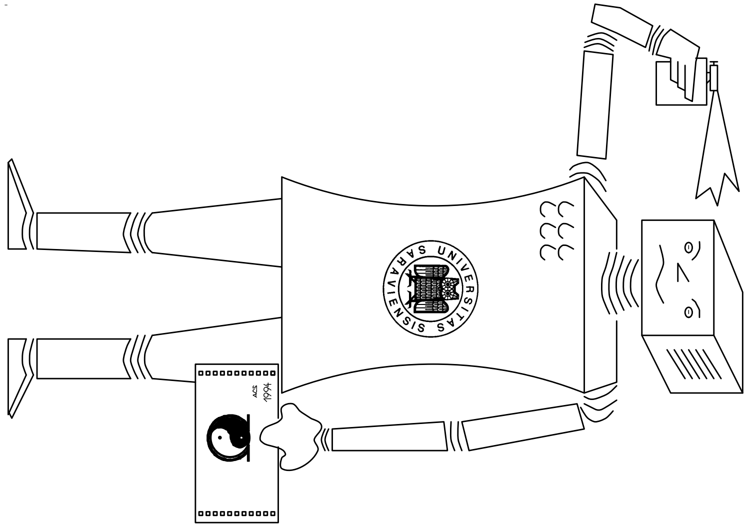
SEKI Report

UNIVERSITÄT DES SAARLANDES
FACHBEREICH INFORMATIK
D-66041 SAARBRÜCKEN
GERMANY
WWW: http://js-sfbsun.cs.uni-sb.de/pub/www/

Human Oriented Proof Presentation: A Reconstructive Approach

Xiaorong Huang

SEKI Report SR-SR-94-07



Human Oriented Proof Presentation: A Reconstructive Approach

Dissertation
zur Erlangung des Grades
des Doktors der Naturwissenschaften
der Technischen Fakultät
der Universität des Saarlandes

von

Xiaorong Huang

Saarbrücken
1994

Tag des Kolloquiums:	17. 05. 1994
Dekan:	Prof. Dr. Günter Hotz
Berichterstatter:	Prof. Dr. Jörg Siekmann
	Prof. Dr. Wolfgang Wahlster

Acknowledgements

The first person I want to thank is my advisor, Jörg Siekmann. He virtually introduced me into the field of artificial intelligence. It is his genuine interest in my work and his enthusiasm that continuously encouraged me throughout my thesis work.

I have been very fortunate to have the opportunity to work within the Ω–MKRP group. I am especially indebted to Christoph Lingenfelder, who explained to me the problem of proof transformation, and integrated me into the group. I am equally indebted to Manfred Kerber. The declarative computational model for human oriented deduction is a collaborative work with him. I wish to thank Dan Nesmith for the helpful discussions on natural deduction proofs, and for his valuable suggestions and prompt help with respect to implementation. Thanks also go to Michael Kohlhase and Erica Melis, for many stimulating discussions. Finally, a very special thanks goes to Armin Fiedler, who implemented part of the system *PROVERB*.

I benefited greatly from the natural language generation group at the University of Saarbrücken. I am greatly indebted to Norbert Reithinger, who not only introduced to me the young field of NL generation, but also served as a most competent advisor throughout my thesis work. A very special thanks is due to Wolfgang Finkler and Anne Kilger, who have generously allowed me to use their generator TAG-GEN, and have always provided prompt support.

I also want to thank Robert Dale, who pointed out to me the difficulty in the segmentation of texts.

Manfred Kerber, Norbert Reithinger, Jörg Siekmann and Wolfgang Wahlster read earlier versions of this thesis, I want to thank them for their exact reading and their constructive suggestions. Michael kohlhase, Dan Nesmith and Armin Fiedler also read part of the thesis.

I have made too many friends in Germany to name them all. But I have to mention some of them who made the last several years not only bearable but also delightful: Ute Ambrosiums and Rolf Socher-Ambrosiums, Gerd Gaberdan, Barbara Schütt-Kerber and Manfred Kerber. The life in a foreign country is always accompanied with homesickness. It has been the friendship with Liu Yu and Yan Weixia, Liu Libin and Chen Fang, Xie Yanan, Wang Mingyi, that has made me feel at home time and again.

My parents have had more of an impact on my life than anyone else. They have supported me unconditionally and encouraged me continuously throughout my life. I wouldn't have finished this thesis without everything they have done for me.

I sincerely thank the Konrad-Adenauer foundation, who has provided the necessary financial support for me to start my study in Germany, and supported me in the first three years of my thesis work. Credit is also given to DFG (Deutsche Forschungsgemeinschaft), who provided financial support for the last three years.

Contents

Chapter 1

Overview: Towards a Reconstructive Approach for Presenting Proofs

Over the past thirty years there have been significant achievements in the field of automated theorem proving with respect to the reasoning power of the inference engines. Although some effort has also been spent to facilitate more user friendliness of the deduction systems, most of them failed to benefit from more recent developments in the related fields of artificial intelligence (AI), such as natural language generation and user modeling. In particular, no model is available which accounts both for human deductive activities *and* for human proof presentation. In this thesis, a reconstructive architecture is suggested which substantially abstracts, reorganizes and finally translates machine-found proofs into natural language. Both the procedures and the intermediate representations of our architecture find their basis in computational models for *informal mathematical reasoning* and for *proof presentation*. User modeling is not incorporated into the current theory, although we plan to do so later.

1.1 Why Proof Presentation?

Let us motivate the need of appropriate proof presentation techniques with the help of a simple example: the example is taken from [Lin90], which is a formulation of part of the subgroup criterion as discussed in [Deu71]. This is the problem:

Problem

Let G be a group, $S \subset G$, if for all x,y in S, $y * x^{-1}$ is also in S, then for every a in S its inverse is also in S.

Most existing automated theorem provers work with very machine oriented formalisms like resolution and without appropriate proof presentation techniques, users of such systems are directly confronted with proofs that are represented in such machine-oriented

The set of initial clauses:

$$C1 = \{+(u * u^{-1} = e)\} \qquad C2 = \{+(e * w = w)\}$$
$$C3 = \{-(x \in S), -(y \in S), -(x * y^{-1} = z), +(z \in S)\}$$
$$C4 = \{+(a \in S)\} \qquad C5 = \{-(a^{-1} \in S)\}$$

The resolution steps:

C3,4 & C3,1:	add R1:	$\{-(\mathrm{x} \in \mathrm{S}), -(\mathrm{y} \in \mathrm{S}), -(\mathrm{x} * \mathrm{y}^{-1} = \mathrm{z}), -(\mathrm{y}' \in \mathrm{S}), -(\mathrm{z} * \mathrm{y}'^{-1} = \mathrm{z}'), +(\mathrm{z}' \in \mathrm{S})\}$
R1,1 & C4,1:	add R2:	$\{-(\mathrm{y} \in \mathrm{S}), -(\mathrm{a} * \mathrm{y}^{-1} = \mathrm{z}), -(\mathrm{y}' \in \mathrm{S}), -(\mathrm{z} * \mathrm{y}'^{-1} = \mathrm{z}', +(\mathrm{z}' \in \mathrm{S})\}$
R2,1 & C4,1:	add R3:	$\{-(\mathrm{a} * \mathrm{a}^{-1} = \mathrm{z}), -(\mathrm{y}' \in \mathrm{S}), -(\mathrm{z} * \mathrm{y}'^{-1} = \mathrm{z}', +(\mathrm{z}' \in \mathrm{S})\}$
R3,2 & C4,1:	add R4:	$\{-(\mathrm{a} * \mathrm{a}^{-1} = \mathrm{z}), -(\mathrm{z} * \mathrm{a}^{-1} = \mathrm{z}'), +(\mathrm{z}' \in \mathrm{S})\}$
R4,1 & C1,1:	add R5:	$\{-(\mathrm{e} * \mathrm{a}^{-1} = \mathrm{z}'), +(\mathrm{z}' \in \mathrm{S})\}$
R5,1 & C2,1:	add R6:	$\{+(\mathrm{a}^{-1} \in \mathrm{S})\}$
R6,1 & C5,1:	add R7:	$\square$

Figure 1.1: The Resolution Proof

NNo	S;D		Formula	Reason
			Definitions	
1.	;1	$\vdash$	$\forall_u\ u * u^{-1} = e$	(L-Def-Inverse)
2.	;2	$\vdash$	$\forall_u\ e * u = u$	(L-Def-Unit)
			The Proof	
3.	;3	$\vdash$	$\forall_{x,y,z} x \in S \wedge y \in S \wedge x * y^{-1} = z \Rightarrow z \in S$	(Hyp)
4.	;4	$\vdash$	$a \in S$	(Hyp)
5.	;3	$\vdash$	$\forall_{y,z} a \in S \wedge y \in S \wedge a * y^{-1} = z \Rightarrow z \in S$	($\forall$E 3)
6.	;3	$\vdash$	$\forall_z a \in S \wedge a \in S \wedge a * a^{-1} = z \Rightarrow z \in S$	($\forall$E 5)
7.	;3	$\vdash$	$a \in S \wedge a \in S \wedge a * a^{-1} = e \Rightarrow e \in S$	($\forall$E 6)
8.	1;	$\vdash$	$a * a^{-1} = e$	($\forall$E 1)
9.	;4	$\vdash$	$a \in S \wedge a \in S$	($\wedge$I 4 4)
10.	1;4	$\vdash$	$a \in S \wedge a \in S \wedge a * a^{-1} = e$	($\wedge$I 9 8)
11.	1;3,4	$\vdash$	$e \in S$	($\Rightarrow$E 10 7)
12.	2;	$\vdash$	$e * a^{-1} = a^{-1}$	($\forall$E 2)
13.	;3	$\vdash$	$\forall_{y,z} e \in S \wedge y \in S \wedge e * y^{-1} = z \Rightarrow z \in S$	($\forall$E 3)
14.	;3	$\vdash$	$\forall_z e \in S \wedge a \in S \wedge e * a^{-1} = z \Rightarrow z \in S$	($\forall$E 13)
15.	;3	$\vdash$	$e \in S \wedge a \in S \wedge e * a^{-1} = a^{-1} \Rightarrow a^{-1} \in S$	($\forall$E 14)
16.	1,2;3,4	$\vdash$	$e \in S \wedge a \in S \wedge e * a^{-1} = a^{-1}$	($\wedge$I 11 4 12)
17.	1,2;3,4	$\vdash$	$a^{-1} \in S$	($\Rightarrow$E 16 15)
18.	1,2;3	$\vdash$	$a \in S \Rightarrow a^{-1} \in S$	($\Rightarrow$I 17)
19.	1,2;3	$\vdash$	$\forall\ x \in S \Rightarrow x^{-1} \in S$	($\forall$I 18)

Figure 1.2: The ND Proof

formalisms. Figure 1.1 is the resolution proof given for the problem in [Lin90].

Although it has become a standard representation within the automated reasoning community, a resolution proof is difficult to follow even for professional mathematicians. Hence techniques have been developed that transform proofs from a machine-oriented formalism into so called *natural deduction* (ND) proofs. Figure 1.2 is an ND proof for the same problem in Figure 1.1, that can be obtained from the resolution proof by the

NNo	S;D		Formula	Reason
			Definitions	
1.	;1	$\vdash$	$\forall_u\ u * u^{-1} = e$	(Def-Inverse)
2.	;2	$\vdash$	$\forall_u\ e * u = u$	(Def-Unit)
			The Proof	
3.	;3	$\vdash$	$\forall_{x,y,z} x \in S \wedge y \in S \wedge x * y^{-1} = z \Rightarrow z \in S$	(Hyp)
4.	;4	$\vdash$	$a \in S$	(Hyp)
5.	1;	$\vdash$	$a * a^{-1} = e$	(Def-Inverse)
6.	1;3,4	$\vdash$	$e \in S$	(3 4 4 5)
7.	2;	$\vdash$	$e * a^{-1} = a^{-1}$	(Def-Unit)
8.	1,2;3,4	$\vdash$	$a^{-1} \in S$	(3 6 4 7)
9.	1,2;3	$\vdash$	$a \in S \Rightarrow a^{-1} \in S$	($\Rightarrow$I 8)
10.	1,2;3	$\vdash$	$\forall\ x \in S \Rightarrow x^{-1} \in S$	($\forall$I 9)

Figure 1.3: The Assertion Level Proof

Proof:

Let a be in S. According to the definition of inverse element, $a * a^{-1} = e$. According to our hypothesis, e in S. $e*a^{-1} = a^{-1}$ according to the definition of unit. Again according to our hypothesis, a^{-1} is in S.

Figure 1.4: The Natural Language Proof

transformation mechanism described in [Lin90].

Note that the quality of machine-translated ND proofs is normally by far not as good as that of the proof in Figure 1.2. Machine-translated proofs can contain hundreds of lines for relatively simple problems. Even neatly written ND proofs, however, are at a level much lower than proofs typically found in a mathematical textbook, although each single step is simple to follow now. The main reason is ND proof lines are exclusively justified by ND inference rules that stand for some simple syntactical manipulations.

In other words, the ND proofs are still far to close to some machine level and hence a new intermediate representation called ND style proofs at the *assertion level* is introduced in this thesis, which contains more meaningful justifications. These are intuitively understood as the application of a definition or a theorem. The ND proof in Figure 1.2 can now be abstracted to the assertion level, as given in Figure 1.3. For example line 5 is justified by the application of the definition of "inverse". Line 6 is now justified by applying the hypothesis given in line 3, using line 4 and 5 as premises. This derivation requires five steps in the original ND proof. The abstraction and subsequent reduction of the size of the proof in this example is not as significant as it is often the case, since the definitions (of inverse and of unit) are deliberately simplified for the convenience of discussion. An example will be shown in this theis where an ND proof of 134 lines is abstracted to a proof of only 16 lines.

If a user wishes to share his proof with his colleagues, the best way is commonly via a proof in natural language. In order to achieve this, appropriate text planning techniques and futher structuring and reductions must be applied. Figure 1.4 is a possible verbalization of the assertion level proof in Figure 1.3.

In short, in order to make automated reasoning systems really useful in a machine-assisted proof development environment like for example the Ω–MKRP system [HKK+92b], appropriate proof presentation techniques are necessary.

1.2 Identifying the Problem

Viewing automated theorem provers as a special kind of expert systems, the problem we are confronted with is very similar to that of the explanation component of an expert system, i.e. of ways of explaining the line of reasoning of the system to an end-user. Methods of various kinds are devised to augment, to prune, or even to transform the trace of reasoning that is obtained inside of the system [Sho76, WS89]. In particular it has also been investigated how domain models can help to explain rules and how explanations can be given at different levels. Explanations produced this way are in general *tightly bound* with the authentic movement of an expert system from the initial data to the conclusion. Although such explanations are apparently appropriate for system developers or knowledge engineers, they do not meet the requirement of a typical end-user. Instead of the original, often obscure line of reasoning, most end-users are more interested in a convincing line of reasoning supporting the conclusion reached by the expert system. In the light of this observation, a new, so-called *reconstructive* paradigm for explanation has emerged in recent years [WT92]. The central idea of this approach is that a distinct knowledge base should be used to reconstruct a new solution based on the original one. In other words, explanation is now viewed as a problem solving process in its own right. By using more general structures and more idealized methods in the knowledge base for explanation, a more user oriented explanation can be generated.

A parallel development can be observed in the explanation mode of an automated reasoning system as well: Most earlier systems produce only a trace of protocols [OS91, McC90]. In contrast to the situation for expert systems, in general there is an additional hurdle for understanding these protocols: Not only can the lines of reasoning be unnatural and obscure, the formalism in which the proofs are encoded is usually extremely machine-oriented. As a consequence, such primitive explanations are only useful for the system developers themselves. To change this situation, a *reconstructive* approach for explanation has been pursued in this field as well, by transforming proofs from machine-oriented formalisms into more natural formalisms [And80, Mil83, Pfe87, Lin90]. As the target formalism, usually a variation of the natural deduction (ND) proof first proposed by G. Gentzen [Gen35] is chosen. Since there is no one-to-one relationship between machine-oriented proofs and natural deduction proofs, the transformation process virtually involves a reproving of the theorem, however by maximally utilizing the information contained in the original proof. Heuristics of various kinds are developed to improve the quality of the target ND proof. For instance, C. Lingenfelder utilizes the topological structures of the refutation graph both to produce more direct proofs as well as to avoid redundancy (by inserting lemmata [Lin90]). Another technique for inserting lemmas is reported in [PN90].

Until now the reconstruction stops here and the resulting ND proofs are passed over to a presentation component, which works primarily by ordering, pruning and augmentation. The first attempt of transforming natural deduction proofs into natural language was made by D. Chester [Che76]. His system EXPOUND is usually characterized as an example of *direct translation*: Although a sophisticated linearization is applied on the input ND proofs, the steps are translated locally in a template driven way. Equipped with more advanced techniques developed in the field of natural language generation, a more coherent translation was obtained by the MUMBLE system of D. McDonald [McD83], in particular, emphasis was laid on the generation of utterances highlighting important global structures of the proofs, as well as utterances mediating between subproofs. A more recent attempt can be found in THINKER [EP93], where different styles of explaining ND proofs are exploited. In short, it was believed that ND proofs can be adequately presented by resorting solely to *ordering*, *pruning*, and *augmentation*.

All these systems suffer from the same problem: The derivations they convey are exclusively at the level of inference rules of the ND calculus. In contrast to informal proofs found in standard mathematical textbooks, such proofs are composed of derivations familiar from elementary logic, where each detail is presented and the focus of attention is on syntactic manipulations rather than on the underlying semantic ideas. As a consequence, natural language proofs thus produced contain too many minute details, which submerge the central line of reasoning. This would be tolerable for the usual toy examples containing less than a dozen of proof lines, however the results are intolerable, when these systems are assigned the task of presenting realistic ND proofs containing hundreds of lines, as is the general case for non-trivial proofs.

1.3 Contributions

The main aim of this thesis is to find a way out of the difficulties discussed in the previous section by carrying the reconstructive approach a step further using a new target proof formalism. The main contributions of this thesis can be summarized briefly as:

- A framework for *informal mathematical reasoning*, cast as an interleaving process of planning and verification.

- The definition of new intermediate representation for proof presentation, based on deductive operators used in informal mathematics, called *assertion level* operators.

- A procedure *reconstructing* assertion level proofs from machine-generated ND proofs. This procedure substantially shortens the input ND proofs by abstracting ND proof segments to atomic assertion level derivations.

- A computational model for presenting assertion level proofs, as a first attempt to integrate standard hierarchical planning and unplanned local organization. The hierarchical planning is schema-based.
- The identification of various reference forms for intermediate conclusions and inference methods, as well as a salience based treatment. A very natural segmentation of the discourse into an attentional hierarchy is given as a consequence of the distinction between planned and unplanned activities.

Below is an example. Given as input an ND proof of 134 lines, our system produces the following proof:

> "Let F be a group and U be a subgroup of F and 1 be a unit element of F and 1_U be a unit element of U. According to the definition of unit element $1_U \in U$. Therefore there is an X, $X \in U$. Now suppose that u_1 is such an X. According to the definition of unit element $u_1 * 1_U = u_1$. Since U is a subgroup of F $U \subset F$. Therefore $u_1 \in F$. Similarly $1_U \in F$ since $1_U \in U$. Since F is a group F is a semigroup. Since $u_1 * 1_U = u_1$ 1_U is a solution of the equation $u_1 * X = u_1$. Since 1 is a unit element of F $u_1 * 1 = u_1$. Since 1 is a unit element of F $1 \in F$. Since $u_1 \in F$ 1 is a solution of the equation $u_1 * X = u_1$. Since F is a group $1_U = 1$ by the uniqueness of solution. This conclusion is independent of the choice of the element u_1."

1.4 Overview

This thesis is divided into three parts. Part I presents computational model for *informal mathematical reasoning*. Apart from the natural deduction inference rules that reflects the primitive human deductive activity, such a model should incorporate more powerful reasoning procedures involved in standard mathematical practice. In particular, reasoning procedures producing proof segments allowing for *atomic* justifications in the presentation process must be identified. By carefully analysing proofs in mathematical textbooks, and also based on previous studies carried out on informal mathematical reasoning, our study has concentrated on a procedure which derives results that, intuitively speaking, follow by applying a previous result, a theorem or a definition (collectively called an *assertion*). Closely related to the concept of the application of an assertion, we also elaborate on domain-specific, logically compound inference rules carrying out the same task. A derivation based on the application of an assertion or a corresponding compound inference rule is called a step at the *assertion level* . The main contribution of part I is to define ND style proofs at the assertion level, which thenserves as the target formalism for the reconstruction.

The presentation of an ND style proof at the assertion level is handled by a computational model for human proof presentation that is abtained in Part II. In this theory, the process of proof presentation is cast as the combination of knowledge based *top-down*

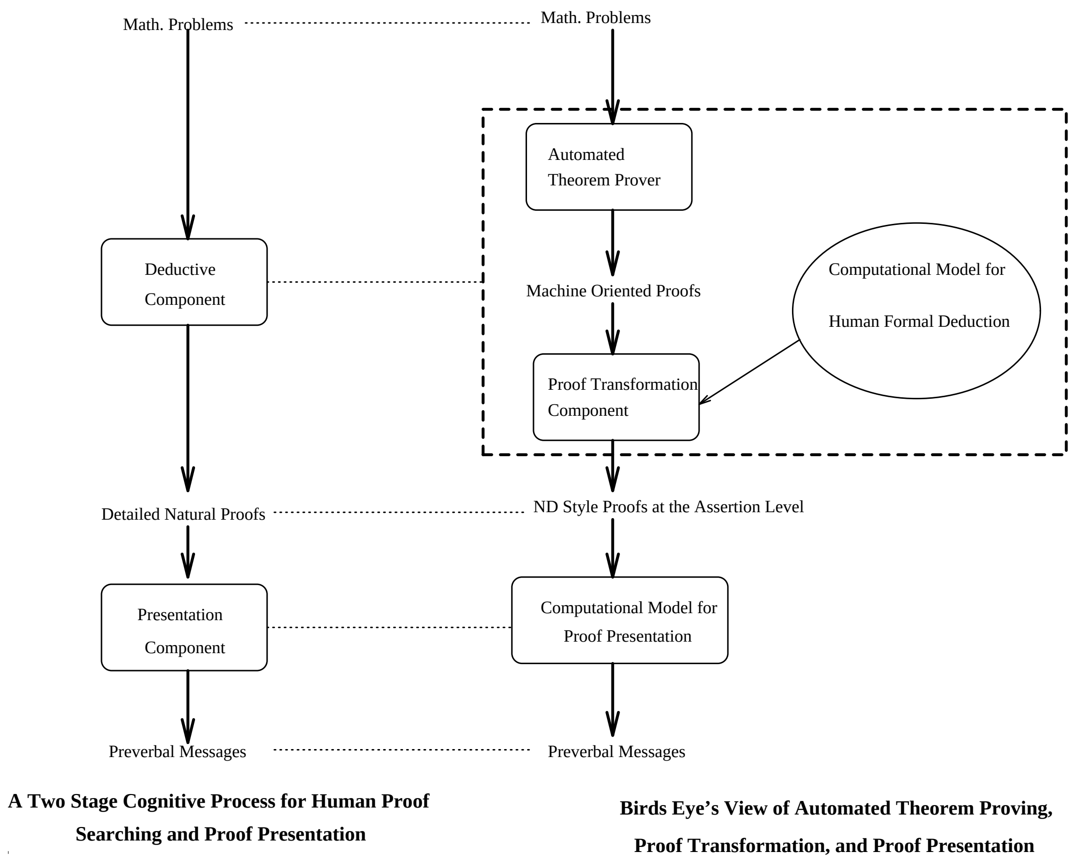


Figure 1.5: A Birds Eye's View of Proof Presentation

planning and focus-guided *bottom-up* presentation. The distinction between top-down planning and bottom-up presentation leads to a very natural segmentation of the proof discourse, that is necessary for a theory addressing problems concerning reference forms.

Part I and Part II together form a coherent theory that accounts for a fairly wide spectrum of human theorem proving and human proof presentation, as illustrated on the left hand side in Figure 1.5. Although in a more full fledged theory, these two cognitive processes should be structured such that they may produce and consume intermediate data in an incremental way, for our purpose, it suffices to model this activity as a relatively independent two-stage process. Note in particular, ND style proofs at the assertion level are supposed to be an appropriate approximation of the product of the human deductive apparatus. In the sequel, these proofs are referred to as *detailed natural proofs* (DNP), as opposed to proofs that have already undergone a presentation process, such that they are usually rearranged and pruned.

Part III gives an overview a prototype of a proof presentation system called *PRO-VERB* (Proof Verbalization) , based on the two computational theories. It is embedded into an interactive proof development environment that is called Ω–MKRP [HKK+92b], a system under development at the University of Saarbrücken. A proof and presentation cycle in Ω–MKRP can be roughly outlined as follows: A resolution based theorem prover is first invoked to solve a mathematical problem. The resulting clausal proof is translated in several steps into an ND style proof at the assertion level. At this point, a presentation component will take over the control and generate a sequence of so called *preverbal messages*, from which a proof in natural language can be produced by an appropriate grammatical treatment. Due to the additional intermediate representation at the assertion level, we are able to generate appropriate proofs in natural language for a broad class of natural deduction proofs. This process is illustrated on the right hand side of Figure 1.5, where dotted lines are used to connect components in our system and their duals in the model of human proof searching and presentation.

Part I

A Computational Model for Informal Mathematical Reasoning

Chapter 2

Introduction

Part I concentrates on a computational model for informal mathematical reasoning, whose aim is to come up with a precise description of the detailed natural proof. As is widely documented, many cognitive processes for solving problems can be modelled within a computational framework of planning [NRL89, New90]. Following this well pursued approach in the field of artificial intelligence, some plan-based theories on deduction have also recently emerged [Bun88, Hut90]. In this thesis, the cognitive process of finding mathematical proofs is cast as an *interleaving process* of planning and verification. From this perspective, a theory of human theorem proving consists primarily of the specification of the *plan operators* at the disposal of a human reasoner, as well as his planning or search strategies.

Because the main concern of this work is a precise description of the output of the human deductive apparatus, our emphasis is mainly on the plan operators. Even here we want to restrict ourselves: Instead of providing a complete description of all possible ways primitive operators can be combined into compound operators, we are interested basically only in primitive operators, and those compound operators lending themselves to *atomic justifications* in the presentation process.

2.1 Previous Work

In this chapter, we first give a brief summary of traditional theories on natural logics and mental reasoning as developed by psychologists and logicians. Subsequently, we report our preliminary empirical study on compound proof structures allowing atomic justifications. Finally, we sketch in an informal way an extension of the traditional framework of mental reasoning, to account for such compound structures.

One of the main streams of psychological theories about mental reasoning has been based on the hypothesis of the existence of a so called *mental logic* or *natural logic*. The word "natural", nevertheless, was first used by the logician Gerhard Gentzen for his logic motivated by the desire "to set up a formal system that came as close as possible to actual reasoning" [Gen35]. More recently, natural logics (also called mental logics) are often studied by cognitive psychologists, as an internal structure that is responsible for the human reasoning competence [Bra78, Lak70, Rip83]. From this point of view, a model for deductive reasoning is believed to contain mainly two components:

1. A logical component containing a repertoire of a deductive vocabulary available to a human being, i.e. a natural logic in form of a set of inference schemata, where the patterns of the premises and of the conclusion are specified in terms of formula schemata.

2. A performance component that contains mainly heuristics and programs responsible for putting inference rules together to form arguments or proofs.

More recently, Johnson-Laird and his colleagues developed a theory using *mental models* to account for human daily reasoning [JL83, JLB90].

2.2 The Main Phenomena

Since we mainly concentrate on the kind of reasoning as encountered in mathematics, we shall nevertheless basically follow the logic based approach. According to such theories, we may predict that the proofs found by the human deductive apparatus can be represented as an ordered graph, where each node contains an intermediate conclusion, whereas the links stand for justifications connecting an intermediate conclusion with its premises. Furthermore, we may predict that the justifications are restricted to the application of primitive operators, i.e. , to the rules of inference in the natural logic. In short, a DNP can be appropriately approximated by a *natural deduction* (ND) style proof.

In order to gain more reliable experience with DNPs, in particular to identify proof structures allowing atomic justifications, the author encoded a number of theorems from different sources into predicate logic. Then the author tried to formulate detailed natural proofs (DNPs) out of the original proofs, by filling back in all reasoning steps that have been omitted in the presentation process. At the same time, it is tried to make explicit the justification for each presented step. This procedure is carried out on six proofs and some exercises in chapter 1 in [Deu71], a mathematical textbook for undergraduates, and some examples discussed in the context of proof planning (two examples about homomorphism in [HKK+92b] and four diagonalization examples in [HKK94]). The same procedure was later repeated on some non-mathematical examples as well (two Portia examples in [Smu78] and the steam-roller example in [Sti86]. To our surprise, only a small amount of the presented steps are justified by inference rules of Gentzen's natural deduction calculus. Most of them are however justified as the application of an axiom, a definition, or a theorem. On the account that mathematical axioms, definitions and theorems are assertions taken as true in a given mathematical context, these justifications will henceforth be collectively called the *application of an assertion*. Besides the examples mentioned above, the characterization of the application of an assertion to be defined in this thesis is checked against numerous proofs in chapter 3 to chapter 5 in [SM77], where applications of certain assertions are explicitly given as justifications. Although not strictly following the standard of psychological empirical studies, this investigation is referred to as our preliminary empirical studies in this work.

Now there are two possible ways for explaining this phenomenon. Firstly we may assume that this phenomenon can be fully ascribed to the presentation process: complex proof segments with certain structural features are always presented as an atomic step, using a justification called the application of a particular assertion. As second alternative however, we may assume a strong correspondence between proof time structure (i.e. proof segments as produced at the time of search by specific reasoning procedures) and atomic steps justified in the presentation process. Since there is enough evidence that mathematicians do plan and verify proofs in terms of the application of definitions or theorems, and in particular, formulate complex search strategies in terms of them, we opt for the second explanation.

One fact supporting this is that domain-specific inference rules at the *assertion level* are widely employed in various theorem proving systems working on natural deduction style proofs. An example is the system EXCHECK [SGBM75, Bla81], a system aimed at supporting *informal* mathematical reasoning. In the system IMPS [FGT92, FGT93], new assertion level inference rules called *macetes* are generated from every definition or theorem in the knowledge base to accelerate the proof search. Although macetes cover only a small part of the ways a human mathematician usually applies an assertion, they relieve the user of a substantial part of applying elementary inference rules. In the system MUSCADET [Pas93], the user may formulate *meta-rules* to generate assertion level inference rules. In the proof development environment Ω–MKRP [HKK$^+$92b], the user can also choose to apply an assertion, apart from the choice of particular natural deduction inference rules.

Once we decide to account for assertion level steps as a phenomenon of reasoning, we are faced with two possibilities again. An assertion level step is either an elementary step in DNP, or a complex proof segment satisfying certain structural constraints. Both cases are possible. Concretely, to account for assertion level reasoning activities, we extend traditional computational models in two ways. Firstly, we add a procedure applying assertions by carrying out a logic level proof conforming to certain structural constraints. Logic level proof segments thus produced are called the *natural expansion* for the corresponding assertion level steps. For instance, given the definition of a subset,

$$\forall_{F,U}\ U \subset F \Leftrightarrow \forall_x\ x \in U \Rightarrow x \in F$$

the natural expansion of the assertion level step deriving $a_1 \in F$ from $U_1 \subset F_1$ and $a_1 \in U_1$ is the logic level proof in Figure 2.1 (i.e. a rather long and tedious proof segment that establishes a fact that would be called “trivial” and hence omitted in a human proof).

Secondly, we assume that there is some mechanism for the *acquisition* of assertion level inference rules. The total sum of inference rules at the disposal of a reasoner is the union of those in natural logic and those acquired. In the sequel, this set of rules is referred to as the *natural calculus*. As will become clear in Part III, only the acquired rules are then used to abstract an ND proofs to an assertion level proof.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\forall_{F,U}\; U \subset F \Leftrightarrow \forall_x\; x \in U \Rightarrow x \in F}{U_1 \subset F_1 \Leftrightarrow \forall_x\; x \in U_1 \Rightarrow x \in F_1}\forall D}{U_1 \subset F_1 \Rightarrow \forall_x\; x \in U_1 \Rightarrow x \in F_1}\Leftrightarrow D, \qquad U_1 \subset F_1}{\forall_x\; x \in U_1 \Rightarrow x \in F_1}\Rightarrow D}{a_1 \in U_1 \Rightarrow a_1 \in F_1}\forall D, \qquad a_1 \in U_1}{a_1 \in F_1}$$

Figure 2.1: An Example of a natural expansion

2.3 Structure of Part I

In chapters 3 to 8, we proceed gradually to our goal: the notion of an *extensible natural calculus*. The chapters are organized as follows: Chapter 3 provides an overall structure of the computational model. Chapter 4 to 7 are each dedicated to one aspect related to the building up of assertional operators: Chapter 4 lays the foundation by specifying a set of primitive inference rules accommodated in this model. Chapter 5 then introduces an operator that generates rules from existing rules by a generalized contraposition. Chapter 6 is devoted to our central topic, namely a procedure drawing conclusions by applying an assertion. Finally, Chapter 7 defines the set of compound inference rules at the assertion level, which can be acquired by chunking and parameterization. Related works and possible applications of the natural calculus are discussed finally in Chapter 8.

Chapter 3

A Framework for the Computational Model

In this chapter, first a general setting of a computational theory accounting for informal mathematical reasoning is set up, as opposed to theories accounting for human daily reasoning. Besides its original motivation to serve the proof presentation process, it is also seen as a first response to Alan Bundy's call for a science of reasoning [Bun90]. This framework is currently realized in a proof development environment Ω–MKRP[HKK$^+$92b], and is deliberately more widely laid out would needed for the proof transformation process alone. In later chapters, however, only those parts that are necessary for our purpose are elaborated upon.

Statically, a reasoner is cast as a knowledge based system, in accordance with the classical view in artificial intelligence. His reasoning competence is exclusively ascribed to his declarative knowledge of various sorts *and* a set of special purpose reasoning procedures. In this first draft of our theory, declarative knowledge includes *rules of inference*, *assertions*, and a collection of *proof schemata*, as well as diverse kinds of *metalevel knowledge*. Since it is the combination of chunks of declarative knowledge and reasoning procedures that constitute the operators to solve various reasoning tasks, these combinations correspond to the notion of *tactics* in earlier literatures [CAB$^+$86, Bun88]. Attached with specifications, which serve as an assessment help in the planning process for a proof, these operators are referred to as *methods*. The total amount of methods constitutes the *basic reasoning repertoire* at the disposal of a human reasoner.

Dynamically, the entire process from the initial analysis of a problem up to the completion of a proof is assumed to be an *interleaving process* of metalevel planning and object level verification. As a central representational structure there is a *proof tree*, which records the current state of the development.

After setting up a general framework for our computational theory in section 3.1, there is an excursion to the notion of method in section 3.2, and a brief description of the dynamic behavior of the interleaving process as a whole in Section 3.3. Readers only interested in proof presentation can skip over these last two sections.

3.1 A Static Description of the Computational Model

As usual we assume the existence of a *long term memory* (LTM) and a short term or *working memory* (WM) , where the latter can be intuitively viewed as the focused part of the entire memory space. It contains only those objects a human being is attending to at present, and it is supposed to be limited in space.

In this section, first the mental objects accommodated in our computational model are categorized, together with the corresponding procedures operating on them.

3.1.1 The Proof Tree

As a theory conceiving theorem proving as an interleaving process of planning and verification, a mental representational structure called a proof tree is used to uniformly accommodate notions like proof sketches, proof plans and finally, proofs themselves. Formally, a proof tree is an ordered tree where each node is labeled by a frame consisting of four slots, represented as:

$$<\mathit{Label},\ \triangle\ \vdash \mathit{Derived\text{-}Formula},\ \mathit{method}>$$

This means that the derived formula is (can or might be, respectively) derived from its children using the indicated method. $\triangle$ stands for the set of hypothesis the derived formula depends on. All of the last two slots of a node may have the value "*unknown*", indicating a proof sketch node. The method slot may also have entry "`plan`", indicating subgoals not yet achieved. Instead of drawing nodes and links explicitly, an extension of the tree structure originally employed by G. Gentzen is adopted for our notion of proof tree. Note that $\triangle$ is normally omitted in this format since it is simply the leaves of the subtree rooted by the corresponding node. The labels are sometimes also omitted. Figure 3.1 is a possible snapshot of the proof tree during the process of proving that the converse relation of a binary relation ρ is symmetric, if ρ is symmetric.

$$\cfrac{\cfrac{\cfrac{[\text{Conv}],\quad [\text{Sym}]}{[1]:\ \text{symmetric}(\rho)}\text{plan},\quad [\text{Conv}],\quad [\text{Sym}]}{[2]:\ \forall_{x,y}\ \langle x,y\rangle \in converse(\rho) \Rightarrow \langle y,x\rangle \in converse(\rho)}\text{plan}}{[3]:\ symmetric(converse(\rho))}\text{Sym}$$

Figure 3.1: An Example of the Proof Tree

Every bar in Figure 3.1 represents a derivation. Nodes separated by commas above a bar are the premises, the node under the bar is called the conclusion. The method slot of the conclusion is put besides the bar. Node [Conv] and [Sym] contain the definitions for converse and for symmetry. Their derived formulas are given separately in 3.1 and 3.2 as follows.

Derived-Formula for [Conv]:

$$\forall_\sigma \ \forall_{x,y} \ \langle x, y \rangle \in \text{converse}(\sigma) \Leftrightarrow \langle y, x \rangle \in \sigma \tag{3.1}$$

Derived-Formula for [Sym]:

$$\forall_\sigma \ \text{symmetric}(\sigma) \Leftrightarrow \forall_{x,y} \ \langle x, y \rangle \in \sigma \Rightarrow \langle y, x \rangle \in \sigma \tag{3.2}$$

Note that one duplication is made of each of them to represent the proof as a tree. The method slot node 1 and 2 is "plan", indicating that they still pending goals. Node 3, in contrast, is justified by applying the definition of symmetry on node 2.

3.1.2 Declarative Knowledge

The following is a listing of the different kinds of declarative knowledge accommodated in our model:

- rules of inference, including a kernel of rules supposedly innate to human beings, usually referred to as the *natural logic* [Bra78, Rip83].
- mathematical formulas collectively called *assertions*, including axioms, definitions, and theorems, interrelated in a certain conceptual structure [Ker91].
- proof schemata, being partial proofs containing metavariables, mainly evolving from proofs previously found.
- metalevel declarative knowledge, specifying possible manipulations at the object level knowledge, in particular proof schemata [HKK92a, Mel93].

Clearly, our theory is a logic based one, built on top of the central hypothesis of the existence of a natural logic. Although there are psychological investigations which argue against this as a general hypothesis [JL83, JLB90], there are others who argue that it is appropriate to assume at least, that the major mode of human mathematical reasoning is based on a logic somewhere situated in the mind.

3.1.3 Procedural Knowledge

Procedures of diverse varieties are incorporated into our computational model:

- Special purpose reasoning procedures:
  - Knowledge interpreters for each type of declarative knowledge mentioned above;

  – An open-ended set of special purpose object level reasoning procedures. The knowledge needed is interwoven into their algorithms;

  Procedures of this type are usually based on a fixed algorithm; neither planning nor heuristic search is involved. These procedures are called as "*basic tools*" by procedures constructing proofs by planning. They correspond to the so-called "effective procedures" discussed by cognitive psychologists (see e.g. [JL83]).

- Autonomous procedures responsible for more complex tasks, such as proof planning or proof checking. Usually, they are animated by relevant intentions, and handle by planning, heuristic searching to fulfill the intentions. It has to be admitted that not much is known yet about these procedures.
- Procedures not involved in the planning process per se, but partially responsible for the increase of the reasoning repertoire. In this work, three of them are addressed: a procedure generates inference rules from existing rules by contraposition, and two generates rules of inference or proof schemata by abstracting from frequently occurring proof segments. The latter two behave similar to perceptive procedures in a theory of general cognition [New90].
- Other procedures, such as the procedure responsible for the comprehension of definitions or problems, as well as bookkeeping procedures (see [Rip83, New90]) moving objects of various data type out of or into memory spaces, and bookkeeping procedures, widely ignored in this study.

As we will see, the inference rules and the assertions, together with their interpreters play a central rule in reconstructing more natural proofs. Before describing them in details, however, an excursion is made in the rest of this chapter into a notion called method. Though rather irrelevant to the issue of the extensible calculus, this notion is central to a theory viewing informal mathematical reasoning as a planning process. A brief description of the dynamic behavior of such a planning process is also given. Users only interested in proof presentation can jump directly to Chapter 4, which introduces the cognitive elementary rules of the natural logic. Chapter 5 proceeds with the first rule generator producing a new rule by contraposition. In Chapter 6 and 7, a procedure that applies assertions and an operator that generates assertion level rules are addressed.

3.2 Methods: The Basic Plan Operators

The concept of a method is central to the reasoning process for proof planing, since methods are the basic units which are planned and executed. The notation of methods is adopted from Alan Bundy, as an extension to the notion of tactics. Intuitively, methods are mathematical problem solving knowledge a system possesses. Apart from general purpose problem solving procedures, in our theory they are in particular designed to capture domain-specific problem solving knowledge, accumulated from past experience.

Viewed within a traditional planning framework, methods play the role of operators. The collection of methods constitutes the basic reasoning repertoire, that is constantly adapted and enriched, as experiences are collected. In the following subsections, first a general definition of the structure of a method is provided, and our definition is compared with similar concepts already introduced in the literature. Then three types of specific methods that we have been able to identify thus far are discussed. In the last subsection, it is illustrated how new methods can be constructed.

3.2.1 General Concepts and Classifications

In our computational model, every method has the following slots:

- *Rating*: A function indicating whether the method is total or partial. It evaluates the appropriateness of applying this method under given circumstances.
- *Precondition*: The preconditions of the problems a method is intended to solve.
- *Postcondition*: The effect the method will end up with.
- *Declarative content*: A piece of declarative knowledge. Currently only three types of object level declarative knowledge are dealt with: the natural deduction inference rules, the assertions (being facts that are either assumed or proved previously), and proof schemata.
- *Procedural content*: Either a standard procedure *interpreting* the piece of declarative knowledge, or a special purpose inference procedure devised for a specific type of problem.

Viewed within a planning framework, the *precondition* and the *postcondition* slots together constitute the logical part of the *specification* of a method, which are both constraints on the partial proof tree. In other words, by these two conditions it is specified whether a method is applicable in a particular proof state or not. If several applicable methods are found the rating procedure should estimate how promising each one is. This concept will not be elaborated upon here, although for real planning tasks this rating may be crucial. For details the reader is referred to [SD93].

It is assumed here, that the planner exclusively consults the specification while planning a proof. The *declarative content* and the *procedural content* slots play the role of a so-called *tactic* in systems like Nuprl [CAB+86] or Bundy's framework for proof planning [Bun90]. Concretely, the declarative content can be an arbitrary piece of declarative knowledge, and the procedural content a Lisp procedure of the following format:

```
interpreter(DeclContent ProofTree &optional other-information)
```

In other words, it is an interpreter which takes as input a piece of declarative knowledge, a pointer to the current proof tree, and optionally other information, and produces a subproof tree that can be integrated into the current partial proof tree. From a logical point of view, the *precondition*, the *postcondition* and the *declarative content* slot together constitute the *declarative part* of a method. These different partitions are illustrated in figure 3.2.

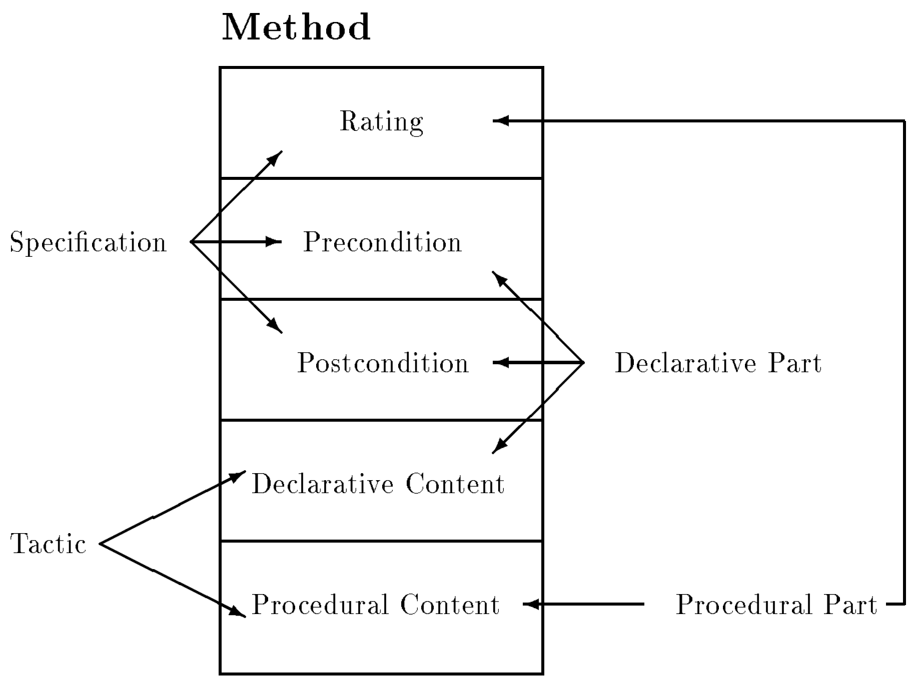


Figure 3.2: The Structure of Methods

Generalizing the concept of a tactic from a procedure (in Bundy's framework) to a pair containing both a procedure and a piece of declarative knowledge is important, since it is now possible to formulate *metamethods* adapting the declarative part of existing methods to suit the problem at hand and thus come up with novel methods [HKK92a, Mel93].

While the power of the special purpose tactics of Bundy rests on the procedural part (the declarative content is empty), the three types of object level methods to be introduced in the next subsection are supported by interpreters, which are standard and simple. Thus our framework is cast so general that it accommodates both a small set of general purpose procedures which operate by interpreting pieces of domain-specific declarative knowledge, and an open-ended set of special purpose reasoning procedures, in which knowledge needed is already implicitly incorporated. This thesis concentrates mainly on three types of general purpose methods at the object level.

3.2.2 Three General Types of Object Level Methods

Within the framework set up so far, three types of object level methods are to be introduced, each handled in a paragraph below. Technically, each type of method is primarily defined by coupling one standard interpreter with chunks of declarative knowledge of one type of object level knowledge. In the sequel, these methods are referred to as methods *applying* a piece of declarative knowledge. To each type of method, also some plausible pre- and postconditions are suggested. The first two types, the application of inference rules and the application of assertions, are cognitively primitive. Their naturalness must be emphasized, which in effect allows us to formulate proof schemata, the third type of object level knowledge, in a quite intuitive way.

3.2.2.1 The Methods Applying Rules of Inference

First there is a procedure that *applies* rules of inference. By inference rules we mean both the elementary rules in the natural logic and compound rules acquired during a reasoning process (compare Chapter 4 and 7).

Now let us turn to the notion of an application of such rules of inference, and their role in the entire process of proof search. It is assumed in this theory that the application of a rule of inference is carried out by a general purpose interpreter which mainly matches formula schemata in rules against formulas contained in support nodes. As a Lisp function, it has the format:

```
rule-interpreter(rule proof-tree &optional other-information)
```

Technically speaking, given a rule of inference of the form:

$$\frac{P_1, \ldots, P_n}{Q} \tag{3.3}$$

the rule interpreter allows the derivation of Q' from $P'_1, \ldots, P'_n$, where Q' and $P'_1, \ldots, P'_n$ are the corresponding instances of Q and $P_1, \ldots, P_n$. Usually, the argument "other-information" points out a set of nodes in the proof tree serving as the support nodes. For rules where the instantiation cannot be determined by the matching alone (for example the "$\forall$D" rule, the instantiation of the universal quantifier), additional information must be provided in the argument "other-information". Now for every rule of inference, there is a method which applies it, since the definition above fully specifies the ability of such methods, and yet is simple enough to be checked without undue efforts. It is plausible to assume that it may be instantiated for every particular rule of inference, and serve as specification in the corresponding methods.

3.2.2.2 Methods Applying Assertions

The second type of important object level knowledge is also common in standard mathematical practice. It concerns objects such as axioms, definitions, lemmas and theorems, and even intermediate results achieved during proof search. They are, in our theory, collectively called *assertions*. Moreover, assertions are normally also interrelated in complex conceptual structures [Ker91]. The notion of the *application* of an assertion, though normally not defined precisely, bears a central role both in the search for proof and in proof documentation. One prima facie evidence is that proofs found by mathematicians are almost exclusively presented in terms of the applications of some assertions.

Let us first illustrate this concept by examining a concrete example of the application of assertions. Given an assertion defining the notion of subset:

$$\forall_{S_1,S_2:Set}\ S_1 \subset S_2 \Leftrightarrow \forall_{x:Element}\ x \in S_1 \Rightarrow x \in S_2$$

We may derive

- $a \in S_2'$ from $a \in S_1'$ and $S_1' \subset S_2'$;
- $S_1' \not\subset S_2'$ from $a \in S_1'$ and $a \notin S_2'$;
- $\forall_{x:Element}\ x \in S_1' \Rightarrow x \in S_2'$ from $S_1' \subset S_2'$.

and so on; by *applying* this definition.

Although introspection seems impossible to reveal the internal structure of the interpreter applying assertions, every application of an assertion can be associated with a proof segment justified by the natural deduction rules only, as is shown in Chapter 6. This proof segment is referred to as its *natural expansion*. A procedure that applies assertions by constructing natural expansions can be found in Chapter 6.

The reasoning ability of a method applying a certain assertion can be captured by a finite set of compound inference rules (Chapter 7). However, it is not plausible to suggest that the planning decisions are based on this information. As a means of assessment, it is apparently too complicated and time consuming. The kind of partial method which seems to be a viable approximation is defined in the following pattern:
Suppose A is an arbitrary assertion, the following is one possible method applying A:

Method: `application-`A	
rating	rating-application-A
pre	exlineset $L \forall_{l \in L}$instance-subformula-neg(formula(l), A)
post	exline n justification(n)=application-A $\wedge$ instance-subformula-neg(formula(n),A)
dec-cont	A
proc	`assertion-interpreter`

Here the predicate instance-subformula-neg(l, A) checks if l is either a subformula of A, an instance thereof, or, thirdly, a negation of the first two cases.

3.2.2.3 Methods Applying Proof Schemata

The third type of methods is tied to a more novel kind of knowledge structure called *proof schema*, and an interpreter *instantiating* it. These notions are introduced in order to partially account for the well-observed phenomenon, that people benefit from their successful and unsuccessful experiences. In other words, with the accumulation of experience, the reasoning ability of a reasoner also evolves. In our theory, this is simulated by the evolution of the collection of proof schemata at the disposal of a reasoner.

Intuitively, proof schemata are proofs or abstract proofs which provide a possible solutions to a reasoning problem. At the very beginning, a proof schema is usually a complete or partial proof found by a reasoning subject for a previous problem. A (partial) specification of the corresponding problem serves as the pre- and postcondition of the method. Undergoing metalevel manipulations, proof schemata also provide solutions to novel problems. These manipulated proof schemata may contain metavariables, which are instantiated by concrete formulas by the procedure applying the proof schemata. Technically, for now, it suffices to understand proof schemata as proof trees containing metalevel variables.

The following method `hom1` is a very simple example of a method applying a proof schema. It represents the following proof idea: If f is a given function, P a defined predicate and the goal is to prove $P(f(c))$, then show $P(c)$ and use this to show $P(f(c))$. The very idea is that f is a homomorphism for the property P and that f can be eliminated (compare [Bun88] for his "ripple-out" tactic).

Method: `hom1`	
rating	`rating-hom1`
pre	(exline 1) $\wedge$ (exline 2)
post	(exline 4)
dec-cont	1. 1; $\vdash$ $\forall_x$ Formula$_f$ (Hyp) 2. 2; $\vdash$ $\forall_x\ P(x) \Leftrightarrow$ Formula$'$ (Hyp) 3. 1,2; $\vdash$ $P(c)$ (PLAN) 4. 1,2; $\vdash$ $P(f(c))$ (PLAN 3)
proc	`schema-interpreter`

The specification of this method means that `hom1` can be applied if the lines 1 and 2 exist in the partial proof under construction and line 4 is an open goal. In this case, the `schema-interpreter` will insert line 3 into the partial proof, as well as adapt the justification of line 4, indicating that 3 is a subgoal.

The formulas in line 1 and 2 are properties of the function f and the predicate P (e.g. their definitions). Both f and P are metavariables standing for (function/predicate) constants of the object logic.

For example, to prove that the converse relation of a binary relation ρ is symmetric (formally: symmetric(converse(ρ))), the method `hom1` can be applied by substituting converse, symmetric, and ρ for the metavariables f, P, and c, respectively. The resulting proof fragment is listed below:

1.	1;	$\vdash$	$\forall_\sigma \forall_{x,y}\ \langle x, y\rangle \in \mathrm{converse}(\sigma) \Leftrightarrow \langle y, x\rangle \in \sigma$	(Hyp)
2.	2;	$\vdash$	$\forall_\sigma\ \mathrm{symmetric}(\sigma) \Leftrightarrow \forall_{x,y}\ \langle x, y\rangle \in \sigma \Rightarrow \langle y, x\rangle \in \sigma$	(Hyp)
3.	1,2;	$\vdash$	$\mathrm{symmetric}(\rho)$	(PLAN)
4.	1,2;	$\vdash$	$\mathrm{symmetric}(\mathrm{converse}(\rho))$	(PLAN 3)

3.2.3 Mechanisms for Constructing Methods

Our theory is also devised to account for the evolution of the reasoner's basic reasoning repertoire. This is achieved by the assumption of *metamethods* and automatic learning procedures. Note that, while the methods cause changes in the current partial proof, metamethods enrich the knowledge base by adding new methods. Metamethods are usually invoked by the intention to solve a specific problem, and their application requires concentration and effort, as opposed to those more perceptual procedures, like remembering a proof or a rule.

3.2.3.1 The Combination of Methods

In this subsection, the automatic part of method construction is briefly described. These activities are very similar to those called *learning during problem solving* described in cognitive models in general, and in frameworks for problem solving in particular, for a comprehensive survey, readers are referred to [Van89].

The most simple form of learning is similar to the process called *compounding* identified in a problem solving process. There, the compounding process puts two or more existing operators into a sequence. A mechanism called *chunking* is proposed in [NR81], which combines compounding with the *tuning* process adapting the heuristic knowledge associated with the operators. In our framework, chunking can be viewed as compounding instantiated methods, adding information about the instantiation into pre- and postcondition.

Methods in our theory may be combined similarly, yet in a somewhat more complicated way. We are not going to go into details here, interested readers are referred to [HKK92a].

3.2.3.2 Metamethods

As already mentioned, our theory is also devised to account for the evolution of a reasoner's basic reasoning repertoire. In addition to those procedures merely remembering useful information, this is achieved mainly through the existence of metamethods manipulating proof schemata. When a reasoner is confronted with a novel yet similar problem, proof schemata evolving from previously successfully found proofs are modified to cope with the new problem. This is also the advice G. Pólya gives in his survey to problem solving [Pól45].

As opposed to methods, metamethods are thought to be very general and problem independent. As a consequence of this it is hoped that meta-metamethods and a whole hierarchy of metalevels are not necessary.

Currently two groups of metamethods have been identified. Guided by heuristic knowledge of different kinds, they will

- *generalize* existing methods built upon a proof schema, or,
- *reformulate* existing methods built upon a proof schema, to suit new problems.

The second kind of metamethod consists of a concrete mapping stated in the declarative content and an interpreter for mappings, which applies a mapping in a controlled way to the logical content of the method to be reformulated. In particular there are strict constraints on mappings to be applied on proof schemata that prohibit the formation of syntactically ill-formed formulas. For details see [HKK92a]. A discussion of reformulations can also be found in [FGT92]. Below, our approach to metamethods will only be illustrated with a *generalization* example.

In section 3.2.2.3 the method `hom1` is introduced, which simplifies a problem by generating an intermediate goal, where a *unary* function symbol is eliminated. Suppose we are facing the problem of proving that the union of two symmetric relations is itself a symmetric relation. What we need is a variant of `hom1`, which is able to handle a *binary* function symbol (i.e. "union") in a similar way.

In the following, it is illustrated how to use the metamethod `add-argument` to obtain a binary version `hom2` from the unary version `hom1`.

Metamethod: `add-argument`	
rating	meta-add-argument-rating
pre	exmethod `M` subterm($f(x)$,post(`M`))
post	goal=post(proc-add-argument(α,`M`))
dec-cont	$\alpha = \{f(x) \mapsto g(x,y)\}$
proc	`proc-add-argument`

This metamethod is supposed to add an argument to a key function f used in a method, and this modified function is called g. Note that the precondition states that there is indeed such a function in `M`. In order to ensure that f is important in `M`, it is required that f occurs in the postcondition of `M`. Based on the mapping given as the declarative content, the procedure `add-argument` modifies the proof schema in `M` by primarily carrying out the following three actions:

- replace all occurrences of terms $f(x)$ by $g(x,y)$ and modify the corresponding quantifications,

- replace all occurrences of terms $f(c)$ by $g(c,d)$ (d has to be a new metavariable standing for a constant),
- if c occurs in a proof line, but not in a term $f(c)$, a copy of this line will be inserted into the proof schema, replacing c by d (in the example below, line 4 is copied from 3).

As a crucial advantage of separating the procedural and the declarative knowledge in methods, the procedural content of `M` can be taken over for the new method.

If we apply `add-argument` to `hom1`, we obtain the new method `hom2`.

Method: `hom2`	
rating	`rating-hom1`
pre	(exline 1) (exline 2)
post	(exline 5)
dec-cont	1. 1; $\vdash$ $\forall_{x,y}$ Formula$_g$ (Hyp) 2. 2; $\vdash$ $\forall_x$ $P(x) \Leftrightarrow$ Formula$'$ (Hyp) 3. 1,2; $\vdash$ $P(c)$ (PLAN) 4. 1,2; $\vdash$ $P(d)$ (PLAN) 5. 1,2; $\vdash$ $P(g(c,d))$ (PLAN 3,4)
proc	`schema-interpreter`

As illustrated in this example, the information of a metamethod is largely encoded as a procedure. It is hoped, however, that this is not a real drawback, since metamethods are devised in a domain independent way, and hopefully no meta-metamethods are needed that would have to reformulate the procedural representation of a metamethod.

3.3 A Dynamic Description of the Computational Model

Since our emphasis is laid more on the static part of the theory, and since not much is yet known about the dynamic behavior, this discussion is tentative and aimed to be suggestive. The basic assumption is, that there is an autonomous procedure responsible for the planning process. The task of this procedure is to analyze the problem, and on the ground of the information contained in methods, to recursively break down the problem into subproblems, and call methods sequentially to solve the subproblems. Since most methods are partial, some of the thus planned proof steps have to be verified subsequently. The current state of such a dynamic development is always reflected in the current proof tree. The planning mode and the verification mode may converge, when the methods employed are complete. Whenever an existing plan fails, a replanning phase usually begins.

There are also metalevel activities. In our theory, a procedure or method is at the metalevel if it causes changes in the knowledge base, rather than the proof tree. Metamethods are usually invoked by the intention to solve a specific problem, and their application require concentration and effort, as opposed to those more perceptual procedures, like the remembrance of a proof or of a rule [HKK92a].

The acquisition of all the three kinds of declarative knowledge can be accounted for: In the first case it is fairly simple. If a particular subproof is carried out repeatedly, its input-output specification may be put together into a new *acquired* rule of inference and remembered. Second, new axioms and definitions are constantly incorporated into the conceptual structure of a reasoner, as well as proved theorems [Ker91]. Third, the initial proof schemata are simply proofs of some problems *learned* by the reasoner, by reading mathematical text books, for instance, or by being taught. The problem specifications can be taken over as the initial methods. Afterwards, new tactics and methods are accumulated, built by metamethods in the attempts to deal with novel problems.

Chapter 4

Cognitively Elementary Inference Rules.

Let us start our approach toward the assertion level inference rules by first introducing a set of cognitively elementary inference rules in this chapter. They play the role of the natural logic studied in cognitive psychology, and are cognitive elementary in the sense that they are assumed to be the elementary internal structure explaining human reasoning competence. The set of elementary inference rules we choose are basically those identified by G. Gentzen in 1930, which comprise his natural deduction calculus [Gen35]. His work is supported once again by our empirical studies of a mathematical textbook showing that the main part of the cognitively elementary rules with respect to logic-based deductive reasoning already had been suggested by him. We basically adopt his calculus NK, plus a pair of symmetric rules handling the connector $\vee$ ($\vee D1$ and $\vee D2$), identified in our empirical study. Because standard many sorted first order logic [Wal87] is assumed as our working language, extensions are also made to handle sorts. The collection of elementary rules adopted into this model is listed in Table 4.1.

Every figure in Table 4.1 represents an inference rule. Formula schemata (formulas containing metavariables) separated by commas above the bar represent preconditions (also called reasons). Metavariables F, G and H can be substituted by any formula, $\forall_{x:S}F_x$, $\exists_{x:S}F_x$ by any formula with $\forall$ or $\exists$ as the top symbol, where "$x : S$" denotes the corresponding bound variable of sort S. $F_{a:S}$ denotes the formula achieved by replacing all occurrences of the bound variable "x" in F_x by an individual constant "a" of sort "S". The metavariable "a" in $\forall D$ can be substituted by an arbitrary term. For rule $\forall I$ and *CHOICE*, in addition, the following *variable conditions* must be checked respectively:

- The variable condition for $\forall I$: metavariable "a" (Eigenvariable in [Gen35]) may not occur in $\forall_{x:S}F_x$ or in any formula in the assumption set $\triangle$.

- The variable condition for *CHOICE*: metavariable "a" may not occur in H, or in any formula in the assumption set $\triangle$.

Structural Gentzen Rules:

$$\frac{}{\triangle, F \vdash F} HYPothesis, \qquad \frac{\triangle, F \vdash G}{\triangle \vdash F \Rightarrow G} DEDuction,$$

$$\frac{\triangle \vdash \exists_{x:S1} Fx, \quad \triangle, F_{a:S2} \vdash H, \quad Subsort(S2, S1)}{\triangle \vdash H} CHOICE,$$

$$\frac{\triangle \vdash F \vee G, \quad \triangle, F \vdash H, \quad \triangle, G \vdash H}{\triangle \vdash H} CASE, \qquad \frac{\triangle, G \vdash \perp}{\triangle \vdash \neg G} IP_1, \qquad \frac{\triangle, \neg G \vdash \perp}{\triangle \vdash G} IP_2$$

Non-Structural Gentzen Rules

$$\frac{\triangle \vdash F, \quad \triangle \vdash G}{\triangle \vdash F \wedge G} \wedge I, \quad \frac{\triangle \vdash F}{\triangle \vdash F \vee G} \vee I_1, \quad \frac{\triangle \vdash G}{\triangle \vdash F \vee G} \vee I_2, \quad \frac{\triangle \vdash F_{a:S}}{\triangle \vdash \forall_{x:S} F_x} \forall I \quad \frac{\triangle \vdash F_{a:S}}{\triangle \vdash \exists_{x:S} F_x} \exists I$$

$$\frac{\triangle \vdash F \wedge G}{\triangle \vdash F} \wedge D_1, \qquad \frac{\triangle \vdash F \wedge G}{\triangle \vdash G} \wedge D_2, \qquad \frac{\triangle \vdash P \vee Q, \quad \triangle \vdash \neg P}{\triangle \vdash Q} \vee D_1,$$

$$\frac{\triangle \vdash P \vee Q, \quad \triangle \vdash \neg Q}{\triangle \vdash P} \vee D_2, \qquad \frac{\triangle \vdash F, \quad \triangle \vdash F \Rightarrow G}{\triangle \vdash G} \Rightarrow D, \qquad \frac{\triangle \vdash \forall_{x:S} F_x}{\triangle \vdash F_{a:S}} \forall D$$

$$\frac{\triangle \vdash F, \quad \triangle \vdash \neg F}{\triangle \vdash \perp} \neg D, \qquad \frac{\triangle \vdash \perp}{\triangle \vdash D} \perp \qquad \frac{\triangle \vdash \neg(\neg F)}{\triangle \vdash F} \neg\neg$$

Table 4.1: Elementary Inference Rules

In the examples, we also adopt the following standard relativization:

$$\forall_{x:S} P \iff \forall_x S(x) \Rightarrow P$$

$$\exists_{x:S} P \iff \exists_x S(x) \wedge P$$

In connection with this, below are three additional rules frequently used for the handling of sorts:

$$\frac{\triangle \vdash \forall_{x:S} P_x, \quad \triangle \vdash S(a)}{\triangle \vdash P_a} \forall D', \qquad \frac{\triangle \vdash S(a) \wedge P_a}{\triangle \vdash \exists_{x:S} P_x} \exists I',$$

$$\frac{\triangle \vdash \exists_{x:S} F_x, \quad \triangle, S(a) \wedge F_a \vdash Q}{\triangle \vdash Q} CHOICE'$$

To enhance the naturalness, the application condition for inference rules are slightly extended. In the traditional natural deduction systems, to apply a rule

$$\frac{P_1, ..., P_n}{Q},$$

for each precondition P_i, a previous proof node must be found, whose derived formula F is the corresponding instance of P_i. Based on our empirical study we have extended it in the following two ways:

1. for each premise P_i, a previous proof node with a derived formula F of the form $... \wedge P_i' \wedge ...$ must be found, where P_i' is the corresponding instance of P_i. With this extension, the first proof below can be simplified to the second proof.

$$\frac{\dfrac{P_1 \wedge P_2}{P_1}\wedge D, \qquad P_1 \Rightarrow Q}{Q} \Rightarrow D; \qquad\qquad \frac{P_1 \wedge P_2, P_1 \Rightarrow Q}{Q} \Rightarrow D$$

2. for each premise P_i of the form $A \Rightarrow B$, a previous proof node with a derived formula B' must be found, which is derived from a node containing as derived formula A'. A' and B' are the corresponding instances of A and B. With this extension, the first proof below can be simplified to the second proof.

$$\frac{\dfrac{\begin{array}{c} A \\ \hline \vdots \\ \hline B \end{array}}{A \Rightarrow B} \Rightarrow I, \quad (A \Rightarrow B) \Rightarrow C}{C} \Rightarrow D; \qquad\qquad \frac{\begin{array}{c} A \\ \hline \vdots \\ \hline B \end{array}, \quad (A \Rightarrow B) \Rightarrow C}{C} \Rightarrow D$$

One thing that is worth mentioning is that the naturalness of the deduction system, with respect to formal reasoning, is largely due to the inference rules under the category of structural rules, where the usual ways of mathematical reasoning are simulated. For example, assumptions are introduced or discharged, problems are divided into cases, etc. The non-structural rules, deal mostly with syntactical manipulations of logical connectives and quantifiers and are rarely found in detailed natural proofs (DNPs).

In traditional theories aimed to account for human deduction, there is a controversy about the origin of such elementary rules [Bra78, Lak87]. It would certainly be an oversimplification to claim that all the rules listed in this chapter are innate. While the non-structural ones are similar to those included in the natural calculus set up by cognitive psychologists [Bra78] (see also Section 8.1) and are therefore more likely to be innate, the structural ones require probably a general training in logic based reasoning, such as the training provided in mathematical courses. From the perspective of cognition, therefore, these are the rules assumed for everyone with a formal training. From a computational perspective, they are elementary in the sense that they are not generated by any operators or procedures.

Chapter 5

Deriving Associated Rules

In this chapter the first of the two operators that are used to generate inference rules is introduced. After a formal definition, some examples will be given. The operator can be viewed as a generalization of the traditional notion of *contraposition* and is specified by the following schema:

If r is a rule in working memory and if it is of the form

$$r = \frac{\triangle \vdash p_1, ..., \triangle \vdash p_n}{\triangle \vdash q} \tag{5.1}$$

then r' (of the form below) is called a rule *associated* with R and can be generated by this operator by *contraposition*:

$$r' = \frac{\triangle \vdash p_1, ..., \triangle \vdash p_{i-1}, \triangle \vdash p_{i+1}, \triangle \vdash p_n, \triangle \vdash \neg q}{\triangle \vdash \neg p_i} \tag{5.2}$$

This implies, intuitively, that the rule r and its associated rules are memorized more or less as one unit of the form

$$\{p_1, ..., p_n, \neg q\} \vdash \bot$$

which means that at least one element of the set $\{p_1, ..., p_n, \neg q\}$ must be false. This derivation, in addition, is carried out in conjunction with the *cancellation of negation*. This means if p_i is of the form $\neg p_i'$, then the conclusion schema in 5.2 will be p_i', instead of $\neg\neg p_i'$. The same holds for formula schema $\neg q$ in 5.2.

Let us now examine some of the rules generated by this operator. The two pairs of rules in 5.3 and 5.4 that were identified in our empirical studies on DNPs can be derived from rule $\Rightarrow D$ and rule $\wedge D$, respectively.

$$\frac{\triangle \vdash a, \triangle \vdash \neg b}{\triangle \vdash \neg(a \Rightarrow b)} \Rightarrow D_1', \qquad\qquad \frac{\triangle \vdash a \Rightarrow b, \triangle \vdash \neg b}{\triangle \vdash \neg a} \Rightarrow D_2' \tag{5.3}$$

$$\frac{\triangle \vdash \neg F}{\triangle \vdash \neg(F \wedge G)} \wedge D'_1 \qquad\qquad \frac{\triangle \vdash \neg G}{\triangle \vdash \neg(F \wedge G)} \wedge D'_2 \tag{5.4}$$

Below is a rule associated with $\forall D$:

$$\frac{\neg P_a}{\neg \forall_x P_x} \forall D'$$

The elementary inference rules introduced in Chapter 3, together with their associated rules, are called the rules at the *logic level*, as opposed to rules generated by the operator to be introduced in the Chapter 7, which will be referred to as being at the *assertion level*.

Although not really effecting the final logical power of the natural calculus, it is worthwhile to point out the existence of various other ways to account for logic level rules generated by this operator, and compare them with the solution we choose from a cognitive perspective. First, to account for the logic level inference rules like those in 5.3 and 5.4, we could simply include them into the set of elementary rules. However, it is firstly quite counter-intuitive to assume a large set of interrelated inference rules as cognitive elementary. The second argument against this solution is that elementary rules and associated rules actually differ in several ways: While the elementary rules are more central, rules associated with them play a comparatively secondary role. In addition, more mental effort is needed to carry out deductions justified by the associated rules. A second proposal seems more plausible, at least at first sight. We may loosen the application condition of inference rules so that an inference rule already has the deductive ability now ascribed to rules associated with it, without assuming the explicit existence of the latter. This solution nevertheless also brings a drawback with it. That is, that the rule applications must now involve search, besides matching, though only in a small space. Based on the above argument, it seems adequate to grant the reasoner the ability of generating associated rules.

Note that this rule generating operator may both be applied to inference rules at the logic level and to domain-specific rules. For instance, given the domain-specific inference rule used in the example in Chapter 7:

$$\frac{U \subset F, a \in U}{a \in F}$$ where "U", "F" and "a" are metavariables for object constant

there are two associated rules which can be generated:

$$\frac{U \subset F, a \notin F}{a \notin U} \qquad \frac{a \in U, a \notin F}{U \not\subset F}$$

In the rest of this chapter, two important properties of this operator useful in subsequent chapters are to be shown. Before doing that, some new notation first. If an inference rule r can be derived from another rule r', then we say that r and r' are associated with each other. Now, let R be an arbitrary set of inference rules, and let $Assoc(R)$ stands for the set of rules associated with at least one rule in R, then:

$$Assoc(R1 \cup R2) = Assoc(R1) \cup Assoc(R2) \tag{5.5}$$

$$Assoc(Assoc(R)) \subseteq R \cup Assoc(R) \tag{5.6}$$

While 5.5 is quite self-evident, 5.6 can also be proved easily by the cancellation of negation. Suppose

$$r = \frac{p_1, \ldots, p_n}{q} \in Assoc(Assoc(R)),$$

then without loss of generality,

$$r' = \frac{q', p_2, \ldots, p_n}{p_1'} \in Assoc(R),$$

where q' equals q'', if q is in form of $\neg q'$; otherwise q' equals $\neg q$. The same holds for p_1'. Now there are two cases:
either

$$r = \frac{p_1, \ldots, p_n}{q} \in R$$

which proves our goal, or, again without loss of generality,

$$\frac{q', p_1, p_3, \ldots, p_n}{p_2'} \in R$$

where p_2' is defined similarly as q', this means

$$r \in Assoc(R)$$

which also proves our goal.

Chapter 6

A Procedure for Applying Assertions

In this chapter, a reasoning procedure central to our model will be introduced, that is, a procedure drawing conclusions by applying assertions. Before going into details, it is worthwhile to pause briefly to clarify the actual situation and the simplifications we have made. Let us first consider the situations within which the procedure under concern may be called. The first case usually occurs during the active proof construction, as already indicated in the introduction. It may as well be called under a more passive circumstance. For instance, during the reading of a mathematical text book, if a reader can not follow an assertion level reasoning step by resorting to a corresponding inference rule acquired previously, it is normal that he comes back to the assertion itself and checks this step by reading the assertion again. This checking process is always accompanied by a reasoning process at the logic level. This logic level proof segment, as already indicated, is called a *natural expansion* (NE) of the corresponding assertion level step.

Although it is possible that the applications of assertions under the two different circumstances are actually carried out by two procedures different from each other in their run-time behavior, a central assumption is made that both of them work by constructing a logic level proof. These logic level proof segments called natural expansions exhibit certain syntactical features. What is provided in this chapter is therefore simply an input-output relation of the procedure. This input-output relation is deduced in Section 6.1 from properties identified in the natural expansions, which appear to characterize them. Only a tentative psychological run-time depiction is attempted in Section 6.2.

6.1 A Characterization of Natural Expansions

In this section, structural constraints of syntactic nature on natural expansions will be employed to define the input-output relation of this primitive reasoning procedure. Even intuitively it is obvious that not all logic level proof segments, in which an assertion is in some way involved, embody an application of this assertion. A trivial example will illustrate this: suppose A is an assertion, B a second arbitrary formula, the valid derivation of $A \wedge B$ does not go in line with our intuition of applying A, no matter what formula A is.

Along with the reconstruction of DNPs discussed in Section 2.2, we have also expanded the reasoning steps at the assertion level to prove segments at the logic level, as natural as possible, with the hope that they will be the natural expansions. Now the characterizations identified thus far will be introduced in two steps in the following two subsections.

6.1.1 Composition and Decomposition Constraints

Before discussing the characteristics in more formal terms, let us first return to the subset example again, used at the outset (compare Section 2.2).

Example 1

In Section 2.2, a proof segment at the logic level is given serving as the NE of a reasoning step at the assertion level, i.e., the derivation of $a_1 \in F_1$ from $U_1 \subset F_1$ and $a_1 \in U_1$ by applying the following assertion:

$$\forall_{F,U}\ U \subset F \Leftrightarrow \forall_x\ x \in U \Rightarrow x \in F$$

The NEs are represented as proof trees, a formalism proposed by Gentzen [Gen35]. The NE of our example is repeated in Figure 6.1 for convenience.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{A}:\ \forall_{F,U}\ U \subset F \Leftrightarrow \forall_x\ x \in U \Rightarrow x \in F}{U_1 \subset F_1 \Leftrightarrow \forall_x\ x \in U_1 \Rightarrow x \in F_1}\forall D[1]}{U_1 \subset F_1 \Rightarrow \forall_x\ x \in U_1 \Rightarrow x \in F_1}\Leftrightarrow D[2],\quad U_1 \subset F_1}{\forall_x\ x \in U_1 \Rightarrow x \in F_1}\Rightarrow D[3]}{a_1 \in U_1 \Rightarrow a_1 \in F_1}\forall D[4],\quad a_1 \in U_1}{a_1 \in F_1}\Rightarrow D[5]$$

Figure 6.1: An Example of a Natural Expansion

Some informal explanations first: In this formalism, every bar represents a reasoning step. The formulas above it, disconnected by commas, and the formula under it are the premises and the conclusion of this step, respectively. It can therefore be concluded that the leaves and the root of such a proof tree are the preconditions and the conclusion of the entire proof segment. Note also, that the steps are numbered, and numbers are put in a pair of square parenthesis. The following properties can be observed in the proof in

Figure 6.1:

1. At every step, there is exactly one premise depending on the assertion $\mathcal{A}$ being applied (the subset definition in this example). This means that there is a certain linear quality in the proof, along the *main branch* from the leaf labeled $\mathcal{A}$ to the root.

2. The premise depending on the assertion being applied, in addition, plays a special role in each step, in that all other premises (if there is any, such as in step 3 and 5), and the conclusion of this step are all either a subformula, or an specialization thereof (F_a is an specialization of both $\forall_x\, F_x$ and $\exists_x\, F_x$). This is henceforth referred to as the *subformula property*.

So the first observation in our empirical studies are that the NE of the application of an assertion is basically a linear sequence of *decompositions*. This is, however, not the whole cake. In other examples, there is also a *composition* process producing the other premises (if there are any) required by the linear decomposition along the main branch. For instance, to apply an assertion of the form $\forall_x\, P_x \wedge Q_x \Rightarrow R_x$, a composition of $P_a \wedge Q_a$ from the two separate premises P_a and Q_a has to be performed before deducing R_a. Let us now formalize the concepts of decomposition and composition, necessary for the formulation of the constraints.

Definition 6.1 An inference rule of the form $\dfrac{\triangle \vdash A, \triangle \vdash P_1, \ldots, \triangle \vdash P_n}{\triangle \vdash Q}$ is a *decomposition* rule with respect to formula schema A, if all applications of it, written as $\dfrac{\triangle \vdash A', \triangle \vdash P'_1, \ldots, \triangle \vdash P'_n}{\triangle \vdash Q'}$, satisfy the following condition: $P'_1, \ldots, P'_n$ and Q' are all either

1. a proper subformula of A', or

2. a specialization of A' or of one of its proper subformula, or,

3. a negation of one of the first two cases.

Under this definition, $\vee D$, $\Rightarrow D$, $\forall D$ are the only elementary decomposition rules (see Chapter 4).

Definition 6.2 An inference rule of the form $\dfrac{\triangle \vdash P_1, \ldots, \triangle \vdash P_n}{\triangle \vdash Q}$ is called a *composition* rule if all applications of it, written as $\dfrac{\triangle \vdash P'_1, \ldots, \triangle \vdash P'_n}{\triangle \vdash Q'}$, satisfy the following condition: $P'_1, \ldots P'_n$ are all either

1. a proper subformula of Q', or

2. a specialization of Q' or of one of its proper subformula, or,

3. a negation of one of the first two cases.

Now we again introduce some new terminologies. In a logic level proof tree serving as an NE of the application of an assertion $\mathcal{A}$, the conclusion formula of the root is called the conclusion of the application, and all the leaves except for the assertion $\mathcal{A}$ itself, are called the premises of the application. Now the complete constraints observed during our preliminary empirical studies are stated in more formal terms in Table 6.1.

Composition and Decomposition Constraints

A logic level proof tree serves as an NE of the application of an assertion $\mathcal{A}$, if it satisfies the following constraints:

1. *quasi-linearity property*: At every proof step, there is at most one premise depending on $\mathcal{A}$.

   It can easily be concluded that all proof nodes depending on $\mathcal{A}$ together form a branch in the proof tree, from the assertion $\mathcal{A}$ to the root. The branch is called the *main branch*. Nodes along this branch are now called the *main intermediate conclusions*. The general structure of a proof tree is illustrated in Figure 6.2, where the main branch is the branch from $\mathcal{A}$ to A_n. The formula $\mathcal{A}, A_1, ..., A_n$ denote the main intermediate conclusions, and $P_{i,1}, ..., P_{i,m_i}$ are the main preconditions for the decomposition step from A_i to A_{i+1}. In other words, the main intermediate conclusions are the only intermediate conclusions depending on $\mathcal{A}$. Furthermore, exactly one of their premises depends on $\mathcal{A}$. We are referring to this *linear order* when we later talk about the previous or subsequent main intermediate conclusions.

2. *decomposition property*: Main intermediate conclusions are justified by decomposition rules $R_1, \ldots, R_n$ with respect to the previous main intermediate conclusion. The inference along the main branch is therefore a linear process of decomposition of the assertion $\mathcal{A}$.

3. *composition property:* Other intermediate conclusions are justified by composition rules with respect to the corresponding intermediate conclusion.

Table 6.1: Composition and Decomposition Constraints

Some very desirable properties can be derived from the constraints above. First, the derived formula of all proof lines is a subformula of the assertion being applied, a specialization thereof, or, in the third case, the negation of one of the first two cases. We will come back to this in Section 6.2. Secondly, if we construct schemata of proof trees by replacing by metavariables all the constant symbols in a proof tree, not originally occurring in the assertion being applied, the total number of the possible tree schemata is finite. This can be proved by the definitions of composition and decomposition rules.

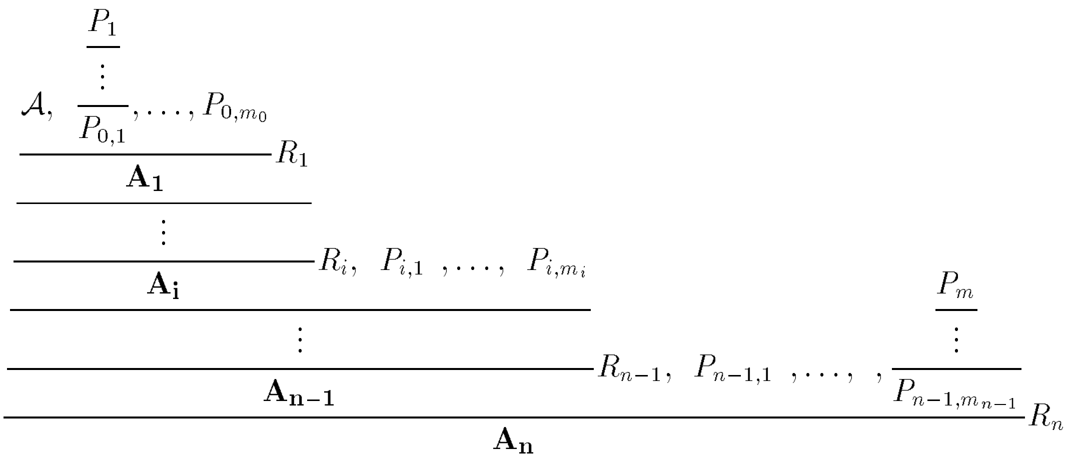


Figure 6.2:

The constants for objects a_1, U_1 and F_1 in Figure 6.1, for instance, can be abstracted to metavariables a, U and F.

6.1.2 Rewriting towards more Natural Formulas

Unfortunately the composition and decomposition characteristics alone are still not enough to cover all the cases encountered in our empirical studies. An exception is shown in the following example:

Example 2

Suppose we have an assertion of the form $A \vee B \Rightarrow C$. Although the derivation of $\neg B$ from $\neg C$ fits fully our intuition of its application, the most natural tree we have constructed in Figure 6.3 does not satisfy the composition and decomposition constraints. One of the "main intermediate conclusions", namely $\neg A \wedge \neg B$, is justified by a rule $\dfrac{\neg(A \vee B)}{\neg A \wedge \neg B}$, which is neither included as a logic level inference rule nor a decomposition rule.

The explanation we have come up with for this phenomenon rests upon the following observation: for formulas of some special patterns there are some other formulas that are logically equivalent but they seem to be more natural for a human being. Once a less natural formula is derived during a proof process as an intermediate conclusion, it is often transformed automatically and implicitly into its more natural equivalent, before the proof proceeds. This is especially likely when motivated by the need of a further composition or decomposition. Therefore, the constraints listed in the previous subsection should be extended to allow this kind of inferences on intermediate conclusions.

The naturalness of a formula is ascribed to the naturalness of it verbalization. This is

$$\cfrac{\cfrac{\cfrac{A \vee B \Rightarrow C, \qquad \neg C}{\neg(A \vee B)}}{\neg A \wedge \neg B}}{\neg B}$$

Figure 6.3:

called the principle of linguistic parsimony by cognitive psychologist studying human daily casual reasoning [JL83]. Table 6.2 is a list of such rewriting inference rules we have found. The first three of them are quite obvious whereas the last three might be controversial. The negation of the compound expressions in the first two pairs, for instance, are found difficult to verbalize in an efficient and straightforward manner. Compared with composition and decomposition constraints discussed in the last subsection, more empirical studies are needed to collect a more complete set of rewriting inference rules. From the perspective of a cognitive theory, we are also still confronted with the problem of explaining the origin and the nature of these rules.

$LessNatural$	$MoreNatural$	$NaturalRewritingRule$
$\neg(A \vee B)$	$\neg A \wedge \neg B$	$\frac{\neg(A \vee B)}{\neg A \wedge \neg B}$
$\neg(A \wedge B)$	$\neg A \vee \neg B$	$\frac{\neg(A \wedge B)}{\neg A \vee \neg B}$
$\neg\neg A$	A	$\frac{\neg\neg A}{A}$
$\neg(A \Rightarrow B)$	$A \wedge \neg B$	$\frac{\neg(A \Rightarrow B)}{A \wedge \neg B}$
$\neg\forall_x\ P_x$	$\exists_x\ \neg P_x$	$\frac{\neg\forall_x\ P_x}{\exists_x\ \neg P_x}$
$\neg\exists_x\ P_x$	$\forall_x\ \neg P_x$	$\frac{\neg\exists_x\ P_x}{\forall_x\ \neg P_x}$

Table 6.2: Rewriting Rules

6.2 A Tentative Description of the Run-time Behavior

What we have proposed thus far is essentially a combination of constraints, which appear to characterize the set of deductions falling under the intuitive concept of the application of an assertion. A precise and detailed run-time description of the procedures is nevertheless still beyond our present knowledge and far beyond the scope of this thesis. We are going, all the same, to at least provide some rough ideas of the possible run-time behavior, based primarily on introspection.

As indicated at the outset of this chapter, the application of an assertion by resorting to a logic level NE may happen during both active proof constructions or passive proof checking. In fact, whenever the corresponding assertion level rule does not reside within working memory and is therefore not at the disposal of the autonomous reasoning procedures, a logic level proof segment is usually constructed, with variation in degrees of detail and explicitness. Let us illustrate this by Schubert's well known steam-roller problem [Sti86], under the circumstance of proof checking.

Example 3

The entire problem is given in natural language:

Wolves, foxes, birds, caterpillars, and snails are animals, and there are some of each of them. Also there are grains, and grains are plants. Every animal either likes to eat all plants or all animals much smaller than itself that like to eat some plants. *Caterpillar and snails are much smaller than birds, which are much smaller than foxes, which in turn are much smaller than wolves. Wolves do not like to eat foxes or grains, while birds like to eat caterpillars and snails like to eat some plants. Therefore there is an animal that likes to eat a grain-eating animal.*

The following is the axiomatization of the third sentence above, where the predicate "E" stands for "eats", "a", "a'", and "p", "p'" are variables of sorts *animal* and *plant* respectively:

Axiom

$$\forall_{a:animal}\ (\forall_{p:plant}\ E(a,p) \vee (\forall_{a':animal}\ a' < a \wedge \exists_{p':plant}\ E(a',p') \Rightarrow E(a,a'))) \tag{6.1}$$

Suppose the derivation of $E(a_1, p_1)$ from $a'_1 < a_1$, $E(a'_1, p'_1)$ and $\neg E(a_1, a'_1)$ is claimed to be justified by applying axiom 6.1. And suppose further that the corresponding assertion level inference rule

$$\frac{a'_1 < a_1, \quad E(a'_1, p'_1), \quad \neg E(a_1, a'_1)}{E(a_1, p_1)} \tag{6.2}$$

where a_1, a'_1, and p_1, p'_1 are metavariables, is either not yet acquired at all or not residing in working memory. The logic level proof in Figure 6.4 may then be constructed, where variables and constants a, a', a_1, a'_1 and p, p_1, p', P'_1 have the sort animal and plant, respectively.

$$
\cfrac{\cfrac{\cfrac{\forall_{a,p}\ E(a,p) \vee (\forall_{a'}\ a' < a \wedge \exists_{p'}\ E(a',p') \Rightarrow E(a,a'))}{\forall_p\ E(a_1,p) \vee (\forall_{a'}\ a' < a_1 \wedge \exists_{p'}\ E(a',p') \Rightarrow E(a_1,a'))}\forall D, \quad \cfrac{\cfrac{a'_1 < a_1, E(a'_1,p'_1), \neg E(a_1,a'_1)}{\neg(a'_1 < a_1 \wedge \exists_{p'}\ E(a'_1,p') \Rightarrow E(a_1,a'_1)}\Rightarrow D'}{\neg\forall_{a'}\ (a' < a_1 \wedge \exists_{p'}\ E(a',p') \Rightarrow E(a_1,a')}\forall D'}{\forall_p\ E(a_1,p)}\vee D}{E(a_1,p_1)}\forall D
$$

Figure 6.4: NE for one Step in Steam Roller Example

Note that the rule $\Rightarrow D'$ and $\forall D'$ used in step 4 and 5 are associated with rule $\Rightarrow D$ and rule $\forall D$ respectively.

Most importantly, by no means do we claim that this logic level NE is actually carried out in run-time by literally writing down the proof above. This proof is much more often or even exclusively followed by reading the assertion being applied by pointing to the corresponding subformula, whose specialization or negation thereof is the intermediate conclusion at this point. It can easily be performed since all the intermediate conclusions of the expanded proof are either specializations of subformulas of the assertion or negations thereof, as is proved in Section 6.1. The process is made still easier, since it can be performed in a quasi-linear way, i.e., along the main branch, on the ground that in most of the practical cases, proofs to establish main premises in a composition manner (represented by subtrees rooted at a main premise) are quite simple. We might even predict the order of steps performed by this implicit reasoning process underlying the check of an assertion level step. A plausible conjecture is that, given the general schema of proof trees in Figure 6.2, the implicit human reasoning proceeds in the following order: the composition of the main preconditions $P'_{1,1}, \ldots, P'_{1,m_1}$, the decomposition from $\mathcal{A}$ to A_1; the composition of the preconditions $P'_{2,1}, \ldots, P'_{2,m_2}$, the decomposition from A_1 to A_2; and so forth. Figure 6.4 is an example. To provide a really procedural description of the application of an assertion is a subject of future research. For more discussion, see Chapter 8.

Before leaving this chapter, now there is the question if the procedure defined above really covers all the cases a human being will intuitively accept as the application of some assertion? This, is obviously an empirical issue. What we may claim is only, that all the reasoning carried out by this procedure seems to be fully in accord with our intuition. No claims can be made nonetheless vice versa, since very possibly our intuitive notion is not all-or-none. For instance, while one may think the derivation of $a \in F \vee b \in F$ from $a \in U \vee b \in U$ and $U \subset F$ is an application of the subset definition, we might as well accept this as a compound process involving first a splitting into two cases and then applying the subset definition twice. Our hope is therefore, that we have at least captured the main part of this notion, although it might have a hazy border.

Chapter 7

Compound Inference Rules at the Assertion Level

In this chapter a structured collection of inference rule schemata at the assertion level is introduced, which on the one hand cover the complete set of inferences carried out by the procedure described in the last chapter, and on the other hand, really have the chance to enter working memory and thus become available to a reasoner. In Section 7.1, possible ways of acquiring assertion level rules are illustrated. Then in Section 7.2, a compact but logically complete tree structure is set up to organize the rule schemata. Examples concerning different types of assertions are used to illustrate this tree structure in Section 7.3. Finally, the cognitive import of this tree structure is then elaborated on in Section 7.4.

7.1 Learning Rules by Chunking and Parameterization

Our computational model postulates two ways for acquiring new assertion level rules. First, since there is evidence that premise-conclusion patterns of repeated actions will be remembered as new operators, we believe that patterns of repeated applications of an assertion may be remembered as new rules. Similar phenomena are called in other systems the learning of *macro-operators* [FHN72], or *chunking* [New90]. As argued in [RL86, Die92], the latter is also be very similar to *explanation-based learning*. On account of this, domain-specific rules are also referred to as compound rules or macro-rules. Secondly, new assertion level rules may also generated as rules associated to assertion level rules already residing in working memory (see Chapter 5). We continue with our subset example to illustrate this.

Example 1 (continued from page 42)

Suppose that the assertion application procedure has just constructed a proof tree as given in Figure 6.1. Our assumption is that possibly the reasoner learns the following inference rule schema as well, apart from merely drawing a concrete conclusion $a_1 \in F_1$ from premises $a_1 \in U_1$ and $U_1 \subset F_1$:

$$\frac{a \in U, U \subset F}{a \in F} \tag{7.1}$$

where a, U and F are metavariables for object terms. More generally, hand in hand with deductive steps supported by proof trees represented in Figure 6.2, a corresponding inference rule taking the form of 7.2 may be acquired.

$$\frac{P'_1, \ldots, P'_m}{A'_n} \tag{7.2}$$

where $P'_1, \ldots, P'_m$ are formula schemata obtained from $P_1, \ldots, P_m$ (the formulas attached to the leaves, except the assertion $\mathcal{A}$ itself) and A'_n is the formula schemata obtained from A_n, attached to the root. The formula schemata are obtained from the corresponding formulas by replacing constant symbols not originally occurring in $\mathcal{A}$ by new metavariables. This is called *parameterization*. Obviously, these constant symbols must occur in formulas serving as premises, such as $a \in U$ and $U \subset F$ in our example.

This learning process can be viewed as a special case of *chunking and variablization* [New90]. It can also be viewed as a special form of explanation-based generalization [MKKC86, RL86, Die92]. See Chapter 8 for details.

Once a new domain-specific rule is derived as described above, it may be used to generate other assertion level rules associated with it. As will be discussed in Section 7.4, there are surely other ways of generating assertion level rules, although it seems reasonable to assume that they are all logically equivalent.

7.2 A Structured Representation

In this section, firstly the set of assertion level inference rules will be pinned down, which enter the working memory either by chunking and parameterization, or as rules associated with existing ones. It will be shown as well, that for each assertion, the total sum of the corresponding inference rule schemata is finite.

Let us first consider assertion level rules generated by chunking and parameterization. Based on the discussion of the last section, there is obviously a one-to-one correspondence between assertion level inference rules and tree schemata, achievable by parameterization. Because there is a subtree relation between such tree schemata, it will be shown that the set of maximal tree schemata can be used to represent the complete set of rules learnable this way. Some further terminologies again. Let $\mathcal{A}$ be an assertion and $\mathcal{B}$ a set of logic level inference rules. Now let $R(\mathcal{A}, \mathcal{B})$ designate the set of inference rules acquired by chunking and parameterizing NEs using exclusively logical level rules in $\mathcal{B}$. It is especially interesting to examine the set of proof tree schemata designated by $Tree(\mathcal{A}, \mathcal{B})$ such that for all $r \in R(\mathcal{A}, \mathcal{B})$, there is a tree schema $t \in Tree(\mathcal{A}, \mathcal{B})$, such that r can be accounted for by a subtree of t. Rule r is called *terminal* if it is accounted for by a tree $t \in Tree(\mathcal{A}, \mathcal{B})$, rather than a proper subtree. Apparently *Tree*$(\mathcal{A},\mathcal{B})$ is unique. Below is a constructive definition:

i) Start with the tree in Figure 7.1(a), which corresponds to the rule $\frac{}{\Delta, A \vdash A}$,

ii) If there is a tree t in the form of Figure 7.1(b), and

- $r = \frac{\Delta \vdash a, \Delta \vdash P_1, \ldots, \Delta \vdash P_n}{\Delta \vdash Q} \in \mathcal{B}$ is a decomposition rule with respect to a, now if there exists a substitution σ, such that $A' = a\sigma$, then extend t to a tree t' of the form of Figure 7.1(c).
- If $\frac{\Delta \vdash P}{\Delta \vdash Q}$ is a natural rewriting rule, a similar extension is made (n=0 in Figure 7.1.(c))

iii) If there is a tree t of the form of Figure 7.1(b), and $r = \frac{\Delta \vdash P'_1, \ldots, \Delta \vdash P'_n}{\Delta \vdash Q} \in \mathcal{B}$ is a composition rule with respect to Q, now if there exists a substitution σ, such that $P = Q\sigma$, then extend t to a tree t' of the form of Figure 7.1(d).

$$\overline{\mathcal{A}} \qquad \dfrac{\dfrac{\mathcal{A},\ldots}{\vdots},\ldots,\dfrac{P,\ldots}{\vdots}}{\dfrac{\vdots,\ldots}{A'}} \qquad \dfrac{\dfrac{\mathcal{A},\ldots}{\vdots},\ldots,\dfrac{P,\ldots}{\vdots}}{\dfrac{\dfrac{\vdots,\ldots}{A', P_1\sigma,\ldots,P_n\sigma}}{Q\sigma}} \qquad \dfrac{\dfrac{\mathcal{A},\ldots}{\vdots},\ldots,\dfrac{\dfrac{P'_1,\ldots,P'_n\sigma}{P},\ldots}{\vdots}}{\dfrac{\vdots,\ldots}{A'}}$$

(a) (b) (c) (d)

Figure 7.1: Construction of Tree Schemata

Some explanations: i) initializes a tree with only one node, corresponding to the initial inference rule $\frac{}{\Delta \vdash A}$, ii) and iii) extend existing trees by decomposing the root or the leaves.

The informations contained in this set is in fact rather redundant, since many rules accounted for by one tree schema are often associated to rules accounted for by another tree schema, in the way described in Chapter 5. This reflects the fact that a rule can either be acquired directly, in a reasoning-and-abstraction manner, or it can be derived as an associated rule of another acquired rule. Let us illustrated this by continuing Example 1.

Example 1 (Continued)

With a rule $\dfrac{a_1 \in U_1, U_1 \subset F_1}{a_1 \in F_1}$ already acquired from the subset definition, supported by the natural expansion (NE) illustrated in Figure 6.1, it is only natural for a human to be able to apply the following associated rule:

$$\frac{a_1 \in U_1, a_1 \notin F_1}{U_1 \not\subset F_1}$$

This, however, has as a matter of fact a corresponding NE of its own:

$$\frac{\dfrac{\forall_{F,U}\ U \subset F \Leftrightarrow \forall_x\ x \in U \Rightarrow x \in F}{U_1 \subset F_1 \Leftrightarrow \forall_x\ x \in U_1 \Rightarrow x \in F_1},\ \dfrac{\dfrac{a_1 \in U_1, a_1 \notin F_1}{\neg(a_1 \in U_1 \Rightarrow a_1 \in F_1)}}{\neg\forall_x\ x \in U_1 \Rightarrow x \in F_1}}{U_1 \not\subset F_1}$$

Figure 7.2: A Rule Applying Definition of Subset

In general, if Fig 7.3.a is the corresponding tree schema for a rule $\frac{c1, c2, b1}{b2}$, acquired from an assertion $\mathcal{A}$, the corresponding tree schema for the associated rule $\frac{b1, \neg b2, c1}{\neg c2}$ can be constructed, using the associated logic level rules, as shown in Figure 7.3.b.

$$\frac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A}{\vdots}}{A'}, \dots, \dfrac{\dfrac{\dfrac{c1, c2, \dots}{Cn}}{\vdots}}{C1}}{B1}, \dots}{\vdots}}{Bm, b1, \dots}}{b2}$$

(a)

$$\frac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{A}}{\vdots}}{A'}, \dots \dfrac{\dfrac{\dfrac{b1, \neg b2, \dots}{\neg Bm}}{\vdots}}{\neg B1}, \dots}{\neg C1}}{\vdots}}{\neg Cn}, c1, \dots}{\neg c2}$$

(b)

Figure 7.3: Natural Expansion for Associated Rules

This argument leads exactly to the following property, that makes a more succinct representation possible:

$$R(\mathcal{A}, Assoc(\mathcal{B}) \cup \mathcal{B}) = R(\mathcal{A}, \mathcal{B}) \cup Assoc(R(\mathcal{A}, \mathcal{B})) \tag{7.3}$$

where $\mathcal{B}$ is an arbitrary set of logic level inference rules. A natural corollary is obtained in conjunction with properties 5.5 and 5.6:

$$Assoc(R(\mathcal{A}, \mathcal{B} \cup Assoc(\mathcal{B}))) \subset R(\mathcal{A}, \mathcal{B} \cup Assoc(\mathcal{B})) \tag{7.4}$$

From a static perspective, this means if all rules associated with elementary rules are already at the disposal of the assertion application procedure discussed in the last chapter, the derivation of associated rules will not bring rules really new at the assertion level. Finally, let Rules($\mathcal{A}$) designate the complete set of inference rules acquirable from an assertion $\mathcal{A}$, either by chunking and parameterization or as an associated rule. If Ele stands for the set of cognitively elementary inference rules available to a reasoner, then

$$\begin{aligned} Rules(\mathcal{A}) =& R(\mathcal{A}, (Ele \cup Assoc(Ele))) \cup Assoc(R(\mathcal{A}, Ele \cup Assoc(Ele))) & \\ =& R(\mathcal{A}, Ele \cup Assoc(Ele)) & \text{by 7.4} \\ =& R(\mathcal{A}, Ele) \cup Assoc(R(\mathcal{A}, Ele)) & \text{by 7.3} \end{aligned}$$

Now we have isolated a subset of the rules, which can be represented by $Tree(\mathcal{A}, Ele)$ in a very compact way that also plays an epistemologically more central role. Thanks to the compactness of the tree form, for almost all intuitively meaningful assertions, $Tree(\mathcal{A}, Ele)$ consists usually of only one or two trees. The special cognitive import of these kernel rules will be discussed in Section 7.4. Note that some logical redundancy still remains in this representation, since there are elementary rules associated with each other. Within the elementary rules we have proposed, for instance, the two $\vee$D rules below

$$\frac{\triangle \vdash P \vee Q, \triangle \vdash \neg P}{\triangle \vdash Q}, \qquad \frac{\triangle \vdash P \vee Q, \triangle \vdash \neg Q}{\triangle \vdash P}$$

are associated with each other.

Now let us finish the discussion of the subset example used throughout Part I by showing its $Tree(\mathcal{A}, \mathcal{NK})$, and a listing of some of the rules contained in $Rules(\mathcal{A})$.

Example 1 (Continued)

As shown in Figure 7.4, two tree schemata are needed since the equivalence $\Leftrightarrow$ is understood as the short hand of the conjunction of two implications and therefore can be decomposed in two different ways. The subset definition is repeated below:

$$\forall_{F,U}\ U \subset F \Leftrightarrow \forall_x\ x \in U \Rightarrow x \in F$$

Table 7.1 is a list of some of the rules in $Rules(\mathcal{A})$. Note that, as we argued above, every subtree containing the subset definition as a leaf corresponds to a rule of inference, if we take the other leaves as preconditions and the root as the conclusion. In other words, only subtrees rooted along the path from the leave which is the assertion being applied

$$\frac{\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{A}: \forall_{F,U}\ U \subset F \Leftrightarrow \forall_x\ x \in U \Rightarrow x \in F}{[a]: U_1 \subset F_1 \Leftrightarrow \forall_x\ x \in U_1 \Rightarrow x \in F_1}}{[b]: U_1 \subset F_1 \Rightarrow \forall_x\ x \in U_1 \Rightarrow x \in F_1}, \quad U_1 \subset F_1}{[c]: \forall_x\ x \in U_1 \Rightarrow x \in F_1}}{[d]: a_1 \in U_1 \Rightarrow a_1 \in F_1}, \quad a_1 \in U_1}{[e]: a_1 \in F_1}$$

a: Tree Schema 1 for the Subset Definition

$$\frac{\dfrac{\dfrac{\mathcal{A}: \forall_{F,U}\ U \subset F \Leftrightarrow \forall_x\ x \in U \Rightarrow x \in F}{[a]: U_1 \subset F_1 \Leftrightarrow \forall_x\ x \in U_1 \Rightarrow x \in F_1}}{[b]: (\forall_x\ x \in U_1 \Rightarrow x \in F_1) \Rightarrow U_1 \subset F_1}, \quad \dfrac{a_1 \in U_1 \Rightarrow a_1 \in F_1}{\forall_x\ x \in U_1 \Rightarrow x \in F_1}}{[c]: U_1 \subset F_1}$$

b: Tree Schema 2 for the Subset Definition

Figure 7.4: Tree Schema for the Subset Definition

to the root, called the *main branch*, are of interest. Figure 7.4.a has five such nodes and Figure 7.4.b has three, namely the length of the main branch. Nodes along the main branch are numbered in Figure 7.4.a and Figure 7.4.b for convenience. Each such subtree represents a rule of inference, directly, and associate rules indirectly.

For instance, rule (1) is directly represented by Figure 7.4.a itself and (2), (3) are associated with rule (1). Rule (4) is represented by a subtree rooted at node [c] in Figure 7.4.a. Rule (5) is represented by the proof tree in Figure 7.4.b itself, which has no associated rules because of its variable condition. Rule (6), the associated rule of the terminal rule (7), is non-terminal (compare the definition for terminal rules at the beginning of this section).

7.3 Examples

Knowledge represented in a logic based language can usually be conceived of as a system of assertions. According to the theory proposed in this thesis, these assertions should directly increase the deductive power of a reasoner. To enhance the modularity, assertions are usually grouped together into theories. Theories, in turn, form a hierarchical structure reflecting the dependency relations [Ker91]. Below, concrete examples of assertion level inference rules in several different theories are provided to demonstrate the growth of a

No.	Inference Rule	Derivation(Tree or Association)
(1)	$\dfrac{\triangle \vdash a_1 \in U_1, \triangle \vdash U_1 \subset F_1}{\triangle \vdash a_1 \in F_1}$	Tree in Figure 7.4.a
(2)	$\dfrac{\triangle \vdash a_1 \notin F_1, \triangle \vdash U_1 \subset F_1}{\triangle \vdash a_1 \notin U_1}$	Associated with (1)
(3)	$\dfrac{\triangle \vdash a_1 \in U_1, \triangle \vdash a_1 \notin F_1}{\triangle \vdash U_1 \not\subset F_1}$	Associated with (1)
(4)	$\dfrac{\triangle \vdash U_1 \subset F_1}{\triangle \vdash \forall_x\ x \in U_1 \Rightarrow x \in F_1}$	subtree in Figure 7.4.a., rooted at node [c]
(5)	$\dfrac{\triangle \vdash a_1 \in U_1 \Rightarrow a_1 \in F_1}{\triangle \vdash U_1 \subset F_1}$ where a_1 does not occur in $\mathcal{A}$	Tree in Figure 7.4.b.
(6)	$\dfrac{\triangle \vdash \forall_x\ x \in U_1 \Rightarrow x \in F_1}{\triangle \vdash U_1 \subset F_1}$	Subtree in Figure 7.4.b., rooted at node [c]
(7)	$\dfrac{\triangle \vdash U_1 \not\subset F_1}{\triangle \vdash \neg\forall_x\ x \in U_1 \Rightarrow x \in F_1}$	Associated with (5)

Table 7.1: Some Inference Rules Associated with the Subset Definition

natural calculus, hand in hand with the growth of a knowledge base.

7.3.1 Assertions at the Logic Level

The growth of the natural calculus can best be observed in mathematical reasoning. Mathematical theories are in general set up hierarchically. Even within one theory, subordinate theorems and lemmas are first proved, thus paving the route leading to a main theorem. In terms of our computational model, this means that along with the development of a mathematical theory, a mathematician working with it is constantly equipped with new, more abstract means of drawing conclusions, i.e., the application of intermediate theorems. To prove a theorem in group theory, for instance, a human may use some theorems of set theory.

On the bottom of the hierarchy of mathematical knowledge, a layer of assertions pertaining directly to logic per se can normally be found. To extend their reasoning repertoire beyond the elementary inference rules directly supported by a calculus like that of Gentzen, mathematicians first prove some new assertions.

For example, once the formula

$$\exists_x\ P_x \lor \exists_x\ Q_x \Leftrightarrow \exists_x\ P_x \lor Q_x$$

is proved as a theorem, the following two inference rules may be added to the collection of inference rules at the disposal of a reasoner:

$$\frac{\exists_x\ P_x \vee \exists_x\ Q_x}{\exists_x\ P_x \vee Q_x}, \qquad \frac{\exists_x\ P_x \vee Q_x}{\exists_x\ P_x \vee \exists_x\ Q_x}$$

Some more examples of logic level assertions are given in Table 7.2.

Logic Tautologies	Inference Rules
$\neg\forall_x\ P_x \Rightarrow \exists_x\ \neg P_x$	$\frac{\neg\forall_x\ P_x}{\exists_x\ \neg P_x}, \frac{\neg\exists_x\ \neg P_x}{\forall_x\ P_x}$
$\forall_x\ \neg P_x \Rightarrow \neg\forall_x\ P_x$	$\frac{\forall_x\ \neg P_x}{\neg\forall_x\ P_x}, \frac{P_a}{\neg\forall_x\ \neg P_x}$
$\neg P_a \Rightarrow \neg\forall_x\ P_x$	$\frac{\neg P_a}{\neg\forall_x\ P_x}$
$\forall_x\ P_x \wedge Q_x \Leftrightarrow \forall_x\ P_x \wedge \forall_x\ Q_x$	$\frac{\forall_x\ P_x \wedge Q_x}{\forall_x\ P_x}, \cdots$
$\forall_x\ P_x \vee \forall_x\ Q_x \Rightarrow \forall_x\ P_x \vee Q_x$	$\frac{\forall_x\ P_x \vee Q_x}{P_a \vee Q_a}, \cdots$
$\exists_x\ P_x \wedge Q_x \Rightarrow \exists_x\ P_x \wedge \exists_x\ Q_x$	$\frac{\exists_x\ P_x \wedge Q_x}{\exists_x\ P_x}$

Table 7.2: Some Logic Level Assertions

Note that this splitting of inference rules into elementary rules and compound rules might resolve a problem troubling cognitive psychologists designing natural logics [Bra78]. To cover their empirical data, their natural logics have to be enriched by inference rules which are by nature obviously compound. In our model, these data can be explained by assuming a knowledge base containing assertional knowledge. It is nevertheless worth noting that it is difficult to set up a criterion to distinguish the elementary rules from logic level rules derived from logic theorems, the border between the two may again be hazy.

7.3.2 Assertions at the Epistemological Level

There is an epistemological level in concept description languages like for example the KL-ONE family [Bra79], that provides a hopefully adequate basis for the construction of conceptual level objects. In a logic based representation language, these are all hidden implicitly in the way of encoding concepts as well as the various relations among them into logical formulas [Hay79]. The usual reasoning inherent to these concept constructing primitives can consequently be carried out by applying the corresponding assertions.

As mentioned in previously, many sorted first order logic is assumed as our working language. In principle, however, the results reported in this thesis can be extended to higher order logic. In this section, let us suppose the full power of higher order logic and let us look at some simple examples. Furthermore, let the two predicates $Property(A, P)$ and $Subsume(A, B)$ mean that P is a property of concept A and concept A subsumes concept B. A possible axiomatization may be:

$$\forall_{A,P}\ Property(A, P) \Leftrightarrow (\forall_x\ A(x) \Rightarrow P(x))$$

$$\forall_{A,B}\ Subsume(A, B) \Rightarrow (\forall_x\ B(x) \Rightarrow A(x))$$

Now the well known inheritance relation can be expressed by the following assertion derivable from the two axioms above:

$$\forall_{A,B,P,x}\ (Property(A, P) \wedge Subsume(A, B) \wedge B(x)) \Rightarrow P(x)$$

which enables us to apply a new inference rule:

$$\frac{Property(A, P), Subsume(A, B), B(x)}{P(x)}$$

Another form of inheritance might be neatly captured by an assertion as well:

$$\forall_{A,B,P}\ (Property(A, P) \wedge Subsume(A, B)) \Rightarrow Property(B, P)$$

which would account for the derivation of "*all students are mortal*" from "*all human beings are mortal*" and "*all students are human beings*".

7.3.3 A Non-Mathematical Example

Thus far, examples of logical assertions are discussed, fundamental for mathematical reasoning. Moreover, the subset example used throughout Part I is considered typical for mathematical reasoning. Schubert's steam-roller problem below should illustrate that similar observations can be made in common sense reasoning as well, provided that logic is used as the representation language.

Example 3 (continued from page 47) The encoding of one of its axioms is repeated below (E stands for Eats).

$$\forall_{a:animal}\ (\forall_{p:plant} E(a, p) \vee (\forall_{a':animal}\ a' < a \wedge \exists_{p':plant}\ E(a', p') \Rightarrow E(a, a')))$$

Although $Tree(\mathcal{A}, \mathcal{NK}))$ consists of two trees in this case, due to the redundancy of the two symmetric $\vee D$ rules, only one is shown in Figure 7.5. Since the two $\vee D$ rules are associated with each other, this tree is complete by itself, see [Hua91]. Some inference rules are listed in Table 7.3 (some metavariables are renamed to improve the readability).

No.	Inference Rules	Derivation(Tree or Association)
(1)	$\dfrac{\triangle \vdash \neg E(a,p), a' < a, E(a',p')}{\triangle \vdash E(a,a')}$	Tree in Figure 7.5.
(2)	$\dfrac{\triangle \vdash a' < a, E(a',p'), \neg E(a,a')}{\triangle \vdash E(a,p)}$	Associated to (1)
(3)	$\dfrac{\triangle \vdash \neg\forall_p\ E(a,p), a' < a, E(a',p')}{\triangle \vdash E(a,a')}$	Subtree in Figure 7.5.
(4)	$\dfrac{\triangle \vdash a' < a, E(a',p'), \neg E(a,a')}{\triangle \vdash \forall_p\ E(a,p)}$	Associated to (3)

Table 7.3: Some Inference Rules for the Steam-Roller Axiom

$$\dfrac{\dfrac{\dfrac{\dfrac{\forall_a\ (\forall_p E(a,p) \vee (\forall_{a'}\ a' < a \wedge \exists_{p'}\ E(a',p') \Rightarrow E(a,a')))}{\forall_p\ E(a_1,p) \vee (\forall_{a'}\ a' < a_1 \wedge \exists_{p'}\ E(a',p') \Rightarrow E(a_1,a'))} \quad \dfrac{\neg E(a_1,p_2)}{\neg\forall_p\ E(a_1,p)}}{\forall_{a'}\ a' < a_1 \wedge \exists_{p'}\ E(a',p') \Rightarrow E(a_1,a')}}{a_1' < a_1 \wedge \exists_{p'}\ E(a_1',p') \Rightarrow E(a_1,a_1')} , \dfrac{a_1' < a_1, \dfrac{E(a_1',p_1)}{\exists_{p'}\ E(a_1',p')}}{a_1' < a_1 \wedge \exists_{p'}\ E(a_1',p')}}{E(a_1,a_1')}$$

Figure 7.5: Tree Schema for the Steam-Roller Example

7.4 Cognitive Status of Assertion Level Inference Rules

Do some of the assertion level rules possess a more significant cognitive status? Are rules in reality stored as an unorganized congregation, or does the tree form structure suggested in Section 7.2 have a cognitive import?

If restricted to the chunking and parameterization view of rule acquisition, most of the answers for the above questions might be negative. As already indicated, however, there is at least one other procedure that introduces inference rules into the working memory: the one carrying out the task of comprehending assertions. The effect of the understanding of an assertion must include the introduction of at least some of the corresponding rules, on the ground that a natural expansion is not always involved at assertion level reasoning, even when an assertion is applied by a reasoner for the first time in a context. It is therefore a plausible conjecture that the understanding process of assertions partially constructs tree schemata containing metavariables later appearing in assertion level inference rules. On the ground that only elementary rules are residing permanently in working memory, (in contrast to associated rules), it can be concluded further that rules accounted for by trees in $Tree(\mathcal{A}, Ele)$ have a chance to be learned first in this comprehension process, since the elementary rules alone are involved in their NEs. Furthermore, since NEs of rules accounted for by subtrees are part of the NEs of the terminal rules, there is really a reason to assume a structure similar to $Tree(\mathcal{A}, Ele)$. Based on the discussion above the division of assertion level rules into those accounted for by $Tree(\mathcal{A}, Ele)$, and the associated rules is not only for the sake of mathematical clarity. Although no precise prediction is possible up to now, it is very likely that the first group of rules are acquired during the comprehension process, while members of the latter are derived form members of the former, when motivated during a reasoning process.

Very likely it is an over-generalization to claim that the comprehension process will always result in the memorization of $Tree(\mathcal{A}, Ele)$, since the situation can be more complicated sometimes. Take the $Tree(\mathcal{A}, Ele)$ of the two assertions in Figure 7.6 for instance, both of them consist of two trees. What is special here is that in both cases, the two trees have a common subtree. It is possible that these common subtrees have a better chance to be memorized, while a derivation of the terminal rules needs to be motivated.

$Tree(\mathcal{A}, \mathcal{NK})$ for $P \vee Q \Rightarrow R$

$$\frac{\mathbf{P} \vee \mathbf{Q} \Rightarrow \mathbf{R}, \dfrac{P}{\mathbf{P} \vee \mathbf{Q}}}{\mathbf{R}}$$

$$\frac{\mathbf{P} \vee \mathbf{Q} \Rightarrow \mathbf{R}, \dfrac{Q}{\mathbf{P} \vee \mathbf{Q}}}{\mathbf{R}}$$

$Tree(\mathcal{A}, \mathcal{NK})$ for $P \Rightarrow Q \wedge R$

$$\frac{\dfrac{\mathbf{P} \Rightarrow \mathbf{Q} \wedge \mathbf{R}, \mathbf{P}}{\mathbf{Q} \wedge \mathbf{R}}}{Q}$$

$$\frac{\dfrac{\mathbf{P} \Rightarrow \mathbf{Q} \wedge \mathbf{R}, \mathbf{P}}{\mathbf{Q} \wedge \mathbf{R}}}{R}$$

Figure 7.6: Possible Common Subtrees

Chapter 8

Related Work and Conclusion

8.1 Related Work

There are two sorts of related works: theories dealing with issues such as natural logics or mental reasoning, and AI applications where these concepts play or will play a major role.

Our set of cognitively elementary inference rules originates directly from the long tradition of the study on natural logics, carried out both by logicians and later by cognitive psychologists. The word *natural* was first used by G. Gentzen [Gen35] to describe his logic. He studied the ways in which mathematical inferences are drawn and decided to design a calculus containing a relatively large number of inference rules which *come as close as possible to actual reasoning*, and thus took an important departure from the calculi in Hilbert's tradition. And more recently, natural logic became an active research topic of cognitive scientists, aiming at discovering internal structures that could account for human reasoning competence. Granted the problems that arise from the assumption of a mental logic [JL83, HHNT86, Lak87], it remains an appealing proposal. Various versions of natural logics are consequently developed for this purpose. While Gentzen's logic can be seen as aiming at a characterization of human formal reasoning, with mathematics as a typical example; the second group of natural logics is either designed to capture the nature of casual daily reasoning [WJL72, JL83], reasoning with English connectives [Bra78] or to serve as a semantic interpretation of natural languages [Lak70]. Since we are primarily interested in logic based reasoning, our set of elementary inference rules is essentially adopted from Gentzen.

There are no doubt problems when theories based on a natural logic are used to explain phenomena observed in everyday reasoning tasks. To overcome this difficulty, Johnson-Laird and his colleagues developed a theory using *mental models* as its internal structure [JL83, JLB90]. In their theory, reasoning is performed as the search for alternative models. A slightly enhanced version of a mental model is recently used in another theory called *verbal reasoning*, where the central processes are fundamentally of linguistic nature (encoding premises and conclusions into mental models) [Pol92]. Although this more semantically oriented theory is evidently gaining importance, its role in mathematical reasoning is presently still unclear.

The central issue of Part I, namely the characterization of reasoning intuitively understood as the application of assertions, is closely related to one of Johnson-Laird's *effective procedures* [JL83], accounting for spontaneous daily reasoning. In spite of the difference of task domains and the basic theoretical assumptions, both studies are confronted with a similar problem and have opted for a similar solution. Given a set of premises, the problem that Johnson-Laird confronts as a cognitive psychologist is how to explain that only a small subset of all the valid conclusions are actually drawn by a human being under ordinary circumstances. We, on the other hand, try to delineate a subset of all the valid inferences intuitively understood as the application of an assertion, given an assertion and some other premises. To this end, both approaches have tried to find constraints that appear to characterize the appropriate subset. The composition and decomposition constraints proposed here are of purely syntactical nature. In contrast, Johnson-Laird's characterization is more of a semantic nature and it may be more superior as a psychological explanation. He first proposed a measure of the information content of formulas, and then postulates that *no conclusion contains less semantic information than the premises on which it is based or fails to express that information more parsimoniously.* This, establishes to a greater extent the meaningfulness of the conclusions drawn. Unfortunately, his measure can neither be extended to predicate logic [JL83, Hin73], nor his criterion can split deductions into units called the applications of single assertions. Comparing the two approaches mathematically, within the realm of propositional logic, the set of meaningful reasoning carried out by Johnson-Laird's procedure is in general more complex, that is, involving more than one application of an assertion. Finally, we want to point out another difference with respect to the general theoretical assumptions. While our discussion is entirely built on top of the assumption of the existence of mental inference rule schemata as an internal structure responsible for deductive competence, this assumption is disputed by Johnson-Laird in general. Despite of his success in explaining daily casual reasoning by means of other internal representations, we believe it remains an appropriate assumption with respect to formal deductive reasoning, where the individuals are usually assumed to have received training in logic. Furthermore, our framework makes possible the accommodation of the compound rules,which help to explain some other observations (see Section 7.4).

Ideas similar to our assertion level reasoning can also be found in several implemented systems. The first such system is EXCHECK, which interprets and checks proofs given by the user in an informal way [SGBM75]. No effort was made nevertheless to delineate this notion precisely and a stronger mechanism is used to check justifications given as the application of an assertion. So-called *macetes* can be generated from proved theorems in IMPS [FGT93], which however covers only a part of the reasoning our procedure in Chapter 6 tries to cover. In addition, frequently a user is forced to split a definition or theorem so that desirable macetes can be built from parts of them satisfying certain syntactic properties. Since macetes are automatically generated for proof search, this restriction is appropriate to cut the search space. In the system MUSCADET [Pas93], the choice of desirable assertion level inference rules for the proof context at hand is encoded in the so-called metarules.

Although the concept of a tactic or a proof plan is gaining influence rapidly, the learning process during proof planning has hitherto not been studied in depth. However, learning of plan scripts or plan schemata has been investigated intensively in the field of machine learning, with interesting results on both similarity based learning and case based learning [Moo90]. The acquisition of assertion level rules resembles more another sort of learning attacked in the literature: building new macros from frequently used operator sequences. In the cognitive architecture SOAR, such macro-building is called *chunking*[New90]. In addition, the chunks constructed are variablized. As described in [RL86, Die92], this kind of macro-building is closely related to explanation-based generalization , [MKKC86], where proof trees serving as explanations are variablized, and intermediate steps are cast away.

8.2 Conclusion and Future Work

Part I deals with issues concerning natural logic and cognitive models for informal logic based deduction. Different from similar research carried out in cognitive psychology, where psychological explanations are the main concern, our study is directly motivated by the practical need of presenting proofs or arguments found by machines to a human user in an appropriate way. By analyzing the blueprint of the entire transformation process, we have isolated explicitly an intermediate representation, called detailed natural proofs (DNP) that possesses the quality of a mathematical abstraction of the actual output of the human deductive apparatus. In order to find a formalism suitable for the encoding of DNPs, preliminary empirical studies on human mathematical proofs are carried out, leading to a new computational model for informal logic based reasoning. The main contribution is the identification of a primitive procedure carrying out the application of assertions, which is added to the traditional models. Hand in hand with this procedure, secondly, an operator is added to account for the acquisition of logically compound inference rules at the assertion level. The extensible set of inference rules, including cognitively elementary inference rules and acquired compound rules alike, is called the extensible natural calculus.

According to this model therefore, the output of a reasoning process (DNPs) are proofs consisting of a sequence of proof steps each justified either by an inference rule, or by the application of an assertion. Since the latter will be presented in the same way as a step achieved by applying an assertion level rule, no distinction needs to be made for our purpose. In part III it will be shown, that the extensible natural calculus alone can be used to reconstruct DNPs from machine found proofs.

The topic of informal logic based reasoning is so rich that our study only indicates a beginning. Among the issues to be addressed in the future, we want to mention first that we need a more semantic oriented explanation for our decomposition and composition constraints, probably along the line of Johnson-Laird. Secondly, refinements with respect to run-time behavior must be made. In particular, a model has to be constructed accounting for human proof comprehension, which provides predictions about whether an assertion level step will be understood by resorting to the assertion application procedure, or by matching against an assertion level inference rule acquired in a previous context. This information will provide more guidance to the decision making procedures in the text planners to be addressed in Part II. Finally, it would certainly be interesting to integrate results on proof planning into such a model [Bun88], and thereby provide new means for structuring DNPs.

Finally, we hope this model may be extended to the sort of reasoning that is beyond the realm of first order predicate logic. An important future research topic is therefore to examine the ideas presented in this paper in domains covered by logics of stronger expressive power, for instance modal logics or logics with sort structures [Pri67, RU71, GG84, SS89, Wei93], as well as logics of higher order [And86].

Part II

A Computational Model for Human Proof Presentation

Chapter 9

Introduction

Part II presents a computational theory simulating the human proof presentation process. Although the psychological reality is not our main concern, the theory is also aimed at shedding some light on the possible mechanisms underlying this human cognitive ability. Secondly this theory serves also a more practical goal, namely, the construction of a proof presentation system which is expected to behave in certain important aspects like real mathematical writers. In fact, an implementation of this theory is used as a module in the system *PROVERB*, to be discussed in Part III. After a short discussion of previous work, the scope of the cognitive activities to be covered by this model is delimited. Then we provide a brief blueprint of this model.

9.1 Previous Work

The past two decades have witnessed a proliferation of different techniques for generating connected texts in natural language. The rapidly growing field of NL generation is represented among others in the recent proceedings of the International Natural Language Generation Workshops [Kem87, PSM91, MMN90, DHRS92] and the European Workshops on Natural Language Generation [ZS88, DMZ90]. Below, only three theories directly relevant to this work are sketched.

9.1.1 A Psycho-linguistic Model for Speech

The overall structure of our computational model is borrowed directly from the psycholinguistic model for speakers proposed by Levelt [Lev89]. In this model, speech is taken as a goal directed activity, producing a sequence of utterances to achieve a certain illocutionary goal. The utterances thus produced are called *speech acts*. According to this model, a human speaker can be conceived of as a *conceptualizer*, a *formulator*, and an *articulator*.

The conceptualizer, in turn, can be divided into a *macroplanner* and a *microplanner*, working together in an incremental way. Macroplanning basically selects information to be expressed, to fulfill some illocutionary intentions, which are special communicative intentions. The output of this phase is assumed to be an ordered sequence of *speech act intentions* (sometimes shortened to speech acts (SA) as well), being messages specified for the intended mood (declarative, interrogative, imperative) and the content. The microplanning phase is responsible for semantic and syntactic decisions, delivering a sequence of *preverbal messages* (PM). For an overview of the conceptualizer, see Figure 9.1. The preverbal messages are passed over to the formulator, which performs grammatical and phonological encodings. Finally, natural language utterances are produced by the articulator.

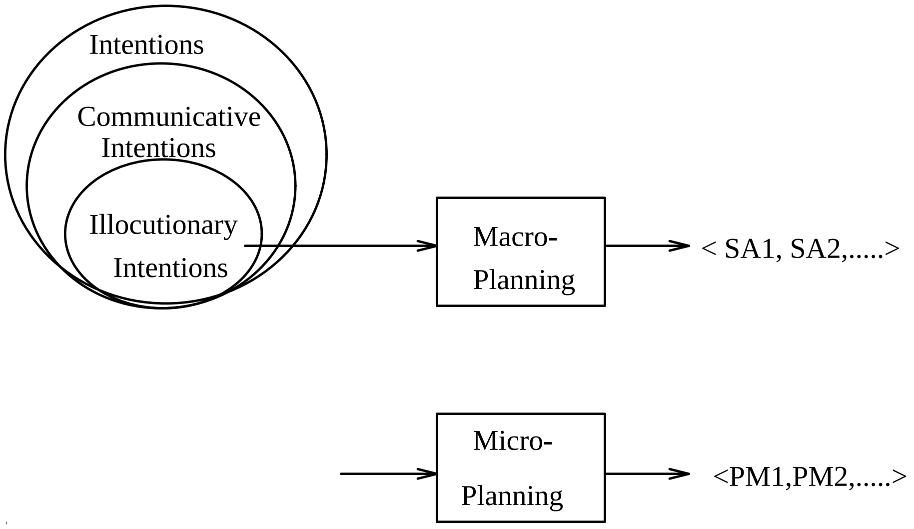


Figure 9.1: From Intention to Preverbal Message ([Lev89])

9.1.2 Schema-Based NL Generation

Mainly two approaches have been pursued for macroplanning in systems that generate coherent multi-sentential text. The first approach relies on script-like structures called *schemata*. For instance, this technique was used to describe complex objects [McK85, Par88]. Schemata are also used to react to object-related misconceptions [McC87].

The schema-based approach finds its basis in the linguistic discussion about *rhetorical predicates* [She26, Gri75b], categorizing the functions of clauses in texts. A further hypothesis is that standard patterns combining rhetorical predicates can be identified. For example, based on her analysis McKeown found that speakers often identify an object or an event among others by:

- identifying the object as a member of some subclass,
- providing constituency or attributive information about the object,

- amplifying the attributives provided
- providing more particular illustrations

Figure 9.2 is the corresponding schema in [McK85], with a sample text generated.

Identification Schema

Identification (class & attribute/function)
analogy/Constituency/Attributive/Renaming/Amplification*
Particular-illustration/Evidence+
Amplification/Analogy/Attributive
Particular illustration/Evidence

Example

"**Eltville** (Germany) 1) An important wine village of the Rheingau region. 2) The vineyards make wines that are emphatically of the Rheingau style, 3) with a considerable weight for a white wine. 4) Taubenberg, Sonnenberg and Langenstuck are among vineyards of note."

Example Classification

1. Identification (class & attribute)
2. Attributive
3. Amplification
4. Particular illustration

Figure 9.2: The Identification Schema

Note that "{}" and "/" indicate optionality and alternative respectively, and that the "+" and "*" indicate the object can appear 1 or more time and 0 or more times, respectively. Since schemata usually contain both options as well as alternatives, the TEXT system of McKeown further employs a focus mechanism to help matching schemata against available knowledge.

9.1.3 RST-Based NL Generation

The main drawback of schema-based planning is its inflexibility. There is a second drawback of this approach if applied to dialog systems: since the *rhetorical predicates* are *compiled* into compound schemata, the intentional relation among parts of a schema has been lost. As a consequence, if an intended effect on the hearer is not achieved, there is no way to allocate the failure. In order to overcome this difficulty, several dialog systems have been developed based on the *rhetorical structure theory* (RST) [Moo89, Rei91].

This theory has also been used to model the effect pragmatic concern may have on the generation of texts [Hov90].

Rhetorical structure theory (RST) [MT87] is a descriptive theory about the organization of natural language texts. Relations are identified which connect parts of texts of various sizes (clauses, sentences or sequences of sentences). An RST relation consists of two parts: a *nucleus* (`N`) and a *satellite* (`S`). The nucleus is more essential to the purpose of the writer. Each RST relation must also specify constraints both on the entities being related, as well as on their combination. The associated effect which a speaker intends to achieve on the hearer is also of central importance. Figure 9.3 is a sample RST relation used in [Moo89].

name: MOTIVATION

constraints on N: present an action in which Hearer is the actor, unrealized with respect to the context of *N*.

constraints on S: none

constraints on N + S combination: Comprehending *S* increases Hearer's desire to perform action presented in `N`.

effect: Hearer's desire to perform action presented in `N` is increased.

Figure 9.3: RST Relation - Motivation

In RST-based text planners the RST theory is operationalized by mapping rhetorical relations identified into corresponding plan operators. Figure 9.4 is the corresponding plan operator constructed from this rhetorical relation. The effect slot contains the goal this operator is designed to achieve. The constraints slot contains the applicability condition of the operator. It also specifies the content to be found by the planner and to be included in the text. More concretely, a set of *goals* must be found such that the *act* is a step toward them. If some such goals are found, the execution of this plan operator will post one or more MOTIVATION subgoals. These goals must be achieved by applying other operators. The operator MOTIVATION is an example of high level operators relating the speaker's goals with rhetorical means. There are also operators which achieve individual rhetorical relations, either by expanding them to lower level relations, or by using primitive speech acts.

9.2 The Scope of this Work

Proof presentation may serve very different purpose. For instance, a reasoner may decide to present a proof in a very detailed way, if he is doing a mathematical examination.

Effect: (PERSUADED ?hearer (GOAL ?hearer (DO ?hearer ?act)))

CONSTRAINTS: (AND (GOAL ?speaker ?goal)
(STEP ?act ?goal)
(GOAL ?hearer ?goal))

NUCLEUS: (FORALL ?goal (MOTIVATION ?act ?goal))

SATELLITES: nil

English Translation:
To achieve the state in which the hearer is persuaded to do an act,
IF the act is a step in achieving some goal(s),
that the hearer shares,
THEN motivate the act in terms of those goal(s)

Figure 9.4: Plan Operator for Persuading User to Do an Act

Under time pressure, it is often irrelevant in which specific order the proof is presented. In contrast, adequate ordering of parts of a proof may play an important role for a lecturer, whose aim is to achieve certain educational goals.

This theory is not going to cover proof presentation under all these circumstances. The circumstance considered here is this: once a reasoner has completed the proof of a problem, he writes it down neatly both as a documentation for himself, and to communicate with others. Our theory does not cover the whole spectrum of all possible presentations. Within the psycho-linguistic model of a human speaker proposed by Levelt, our theory covers first the entire macroplanner for proof presentation. Among the topics concerning the microplanner, only the reference choice is extensively dealt with, with the help of a discourse theory. Other topics such as paragraph planning, sentence planning and lexical choice are treated in a simplistic way to support a running system. The formulator is not treated at all. In the system *PROVERB*, its task is carried out by a tactic component based on the TAG formalism [Kil94].

9.3 The Overview of Our Model

The overall structure of our computational model for proof presentation is illustrated in Figure 9.5. Note that the speech acts (SAs) are called *proof communicative acts* (PCAs) in our domain of application. Below, the components and the intermediate representations are briefly sketched out.

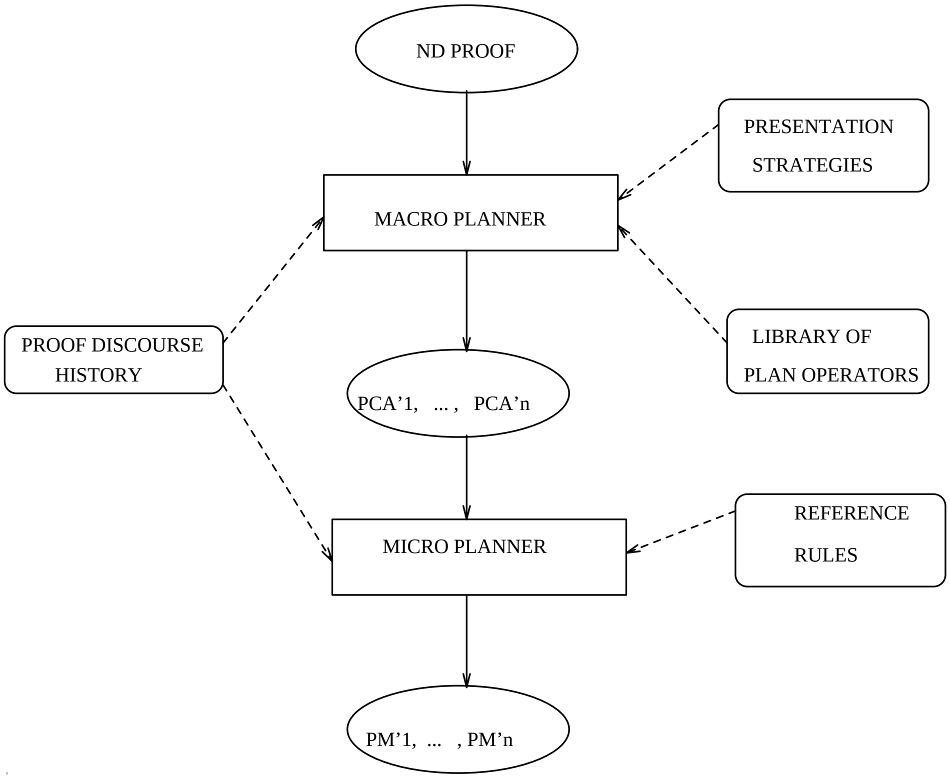


Figure 9.5: Architecture

Input

As shown in Figure 9.5, the presentation model requires as *input* a natural deduction (ND) style proof at the assertion level. As discussed in Part I, it is a proof tree containing mostly assertion level justifications. An assertion level justification is the application of either a definition, a theorem, or a previously proved lemma. Within the entire process of proof searching and proof presentation, ND style proofs at the assertion level are supposed to be the output of the human deductive apparatus. Figure 9.6 is an example of a possible input proof, where labels of some nodes are omitted.

$$\frac{\dfrac{u(u_1, 1_u, *),\ u_1 \in U}{[2]:\ u_1 * 1_u = u_1}\text{Du},\quad \dfrac{u_1 \in U,\ \dfrac{sgr(U,F)}{U \subset F}\text{Dsubgr}}{[3]:\ u_1 \in F}\text{Ds},\quad \dfrac{U \subset F,\ \dfrac{u(U, 1_u, *)}{1_u \in U}\text{Du}}{[4]:\ 1_u \in F}\text{Ds},\quad \dfrac{gr(F,*)}{segr(F,*)}\text{Dg},}{[1]:\ Solution(u_1, u_1, 1_u, F, *)}\text{Tsol}$$

Figure 9.6: An Example Input Proof

The predicates "$segr$", "sgr", "gr", and "u" stand for "semigroup", "subgroup", "group" and "unit element" respectively. "$Solution(a, b, c, F, *)$" should be read as "c is a solution of the equation $a * x = b$ in F." The justifications "Du", "Dsubgr", "Ds", "Dg", and "Tsol" stand for "the definition of unit element", "the definition of subgroup", "the definition of subset", "the definition of group", and "the theorem about solution", respectively.

Apart from the reasoning steps, the proof is augmented by additional information about the proof searching process, in particular an ordered list of nodes, as the subproof structure of the proof. For instance, the proof in Figure 9.6 is associated with the following list:

[2], [3], [4], [1]

indicating that subtrees rooted at the nodes [2], [3], [4], and [1] have been planned as subproofs, and are proved in the above order.

The Proof Discourse History

The dynamic unfolding of a proof discourse is represented in the *proof discourse history* (PDH). This consists of both an internal discourse semantic representation of the text generated thus far, as well as a plan constructed for further presentation. In essence, the former can be conceived as a tree structure reflecting the current state of the ongoing proof.

A Two-Stage Process

As in the model proposed by Levelt, this theory also distinguishes between the *macro-* and *microplanning* stages.

Macroplanning

The macroplanner basically elaborates on communicative goals, selects and orders pieces of information to fulfill these goals. The output is an ordered sequence of *proof communicative acts intentions*, shortened to proof communicative acts (PCAs) in the sequel. Compared with the model of Levelt, PCAs are speech acts in our domain of application. Because our microplanning is restricted to the treatment of reference choices, our PCAs look very similar to preverbal messages (PM), with the reference form information still missing. Below is an example of a PCA (the corresponding PM and a verbalization can be found at the end of this chapter):

```
(Derive Reasons: ((ELE a U)(SUBSET U F))
        Conclusion: (ELE a F)
        Justification: (Def-Subset))
```

The overall framework of our macroplanner can be viewed as a schema-based hierarchical planner. On the one hand, the overall mechanism is very much like the planners in RST-based generators. On the other hand, many of our plan operators contain complex compiled schemata similar to those used in the schema-based approach, because the presentation of many complex proof structures is highly standardized.

As the second important feature, RST-based *top-down* planning is enhanced by a focus guided *bottom-up* presentation. This reflects the observation that language can be both planned and unplanned [Och79, Sib90]. Our theory is therefore a first attempt to integrate hierarchical planning and unplanned organization in a uniform framework. The top-down planning is usually based on explicit presentation knowledge associated with the global structure of the current subproof to be presented. Characteristically, top-down planning requires global information, and involves usually splitting of presentation goals into subgoals. As a consequence, top-down planning is accompanied by global updates of the attentional structure, since it requires both effort and concentration. In contrast, bottom-up presentation is devised to simulate stepwise presentation based on local information. In this mode, the local attention is guided to a next proof node by semantic objects in the local focus. Here, only one node is picked out for presentation, and no explicit subgoal is posted. Both kinds of top-down and bottom-up presentation knowledge are encoded as presentation operators.

Microplanning

Our microplanner is restricted to the treatment of the *reference choices* for the inference methods and for the previously presented intermediate conclusions. While the former depends on static preferences relating to the domain knowledge, the latter is very similar to subsequent references, and therefore sensitive to the context, in particular to its segmentation into an attentional hierarchy. Such a segmentation can be obtained for free in our model by distinguishing between the top-down planning and the bottom-up presentation. While every top-down planning step creates new attentional spaces by posting new subgoals, subsequent bottom-up presentation steps constitute an elementary attentional unit.

Output

The output of the model is a sequence of so called *preverbal messages*, each being a complete specification for one natural language sentence. Below is an example.

```
(Derive Reasons: ((((ELE a U) explicit))
                  ((SUBSET U F) omit))
        Conclusion: (ELE a F)
        Method: (Def-Subset omit))
```

With the help of a dictionary, the tactical component of *PROVERB* produces the utterance:

"Since a is an element of U, a is an element of F."

The Natural Language Proof
(1)Let F be a group and let U be a subgroup of F and let 1 be a unit element of F and let 1_U be a unit element of U. (2)According to the definition of unit element, $1_U \in U$. (3)Therefore there is an X, $X \in U$. (4)Now suppose that u_1 is such an X. (5)According to the definition of unit element, $u_1 * 1_U = u_1$. (6)Since U is a subgroup of F, $U \subset F$. (7)Therefore $1_U \in F$. (8)Similarly $u_1 \in F$, since $u_1 \in U$. (9)Since F is a group, F is a semigroup. (10)Since $u_1 * 1_U = u_1$, 1_U is a solution of the equation $u_1 * X = u_1$. (11)Since 1 is a unit element of F, $u_1 * 1 = u_1$. (12)Since 1 is a unit element of F, $1 \in F$. (13)Since $u_1 \in F$, 1 is a solution of the equation $u_1 * X = u_1$. (14)Since F is a group, $1_U = 1$ by the uniqueness of solution. (15)This conclusion is independent of the choice of the element u_1.

Figure 9.7: A NL Proof Generated by *PROVERB*

9.4 Structure of Part II

The following chapter describes the proof discourse history (PDH), its segmentation into an attentional hierarchy, as well as the proof communicative acts (PCAs) incorporated into our model. Chapter 11 and Chapter 12 are devoted to the macroplanner and microplanner, respectively. This computational model is implemented in the system *PROVERB* to be introduced in Part III, Chapter 13 and finally an outline of future work can be found in Chapter 14.

Chapter 10

The Presentation History and the Proof Communicative Acts

10.1 The Basic Structure of the Presentation History

In our computational theory of proof presentation, a *presentation history* is the central data structure reflecting the dynamic development of the presentation process. The writer's presentation history is the total sum of the information about the argumentative discourse he is currently involved in. Concretely the presentation history comprises the following items:

1. The *name* of the proof.

2. The *hypothesis* of the proof.

3. The *conclusion* of the proof.

4. The linearized sequence of preverbal messages *PMs* generated thus far in the discourse

5. A *proof discourse model*: a model of that part of the proof which the writer believes is already conveyed to the reader. This part corresponds to what usually is called a discourse record or discourse history. In our theory, this model is represented as a partial proof tree similar to that introduced in Chapter 3, containing conveyed intermediate conclusions together with the justification relations.

6. The *rest part* of the proof which has not been presented yet. Since this part of the proof and the part covered in the proof discourse model together constitute an entire proof of the problem, they are implemented as two disjoint parts of one proof tree, with nodes labelled as *conveyed* or *unconveyed*, respectively. Note that upon the completion of the presentation the proof model is not necessarily the original input

proof, since it is affected by both reordering and simplification of the presentation process.

7. A stack of *planned presentation activities*, waiting to be executed. In this model, this stack is not kept explicitly, but simply realized with the help of the recursion management mechanism of the Common Lisp language. Planned activities are only shared partially by the reader, depending on the explicitness of how the goals are articulated. In other words, if goals to be proved are explicitly conveyed, the reader is expected to be aware of these pending goals. In the current version of our theory, this sharedness is not represented.

Before proceeding, let us compare our proof discourse history and theories concerning discourse history in general. While the other items are specific for argumentative texts, the notion of a discourse model is well studied for discourses in general. The discourse model is usually assumed to be the writer's record of what he believes to be shared knowledge about the content of the discourse as it evolves. More precisely, it contains primarily entities introduced by the interlocutors (in our case, the reader is not expected to make any utterances actively), and the predications made about these entities. A further assumption is that the predications about a specific entity are grouped together and stored under the address of that entity in the discourse model [Rei91, Dal92]. Since our computational model handles mainly decisions involving whole assertions as elementary elements, our discourse model is not specified at the level of semantic elements and the relations holding among them. On the contrary, the proof discourse history contains primarily a proof tree, being a set of intermediate conclusions connected by the justification relation. As will be shown, the decisions about the way of unfolding an ongoing partial proof and the way in which the steps are to be conveyed are sensitive to the history of the up to now presented partial proof.

10.2 The Attentional Status of Constituents in the Presentation Model

Apparently both writer and reader are *resource-bounded*, which determines the limitedness of their attentional span. During the process of the presentation of a particular proof, the writer's attention is usually focused on a small portion of the proof. This focusing process is then followed by the reader during successful communication, as far as the conveyed part of the proof is concerned. It is this shared part of the focus structure which is to be dealt with in this section. In Chapter 12, it will be illustrated how this attentional structure plays a crucial role in deciding reference forms in PCAs.

10.2.1 Previous Work

The existence of such focused context spaces is not only intuitively plausible (since our working memory is assumed to be limited [Pos89, LN77]), but is also evidenced by the study of surface linguistic cue phrases and anaphora [Rei85, GS86].

All existing theories agree that the change of focus status does not happen only in terms of a single utterance, but also in terms of larger granularity. These discourse spaces can contain subordinate discourse spaces and thus form an attentional hierarchy [Rei85, GS86]. Any theory modelling the dynamic shifting of the focus of attention must therefore specify the basic units of attention, usually called *discourse spaces*, the boundaries between discourse spaces, the relationship between them, as well as the mechanism which accounts for the dynamic shifting of attention. Despite the intuitiveness of these notions, the segmentation of discourses has proved to be fairly difficult. To pin down the boundary between discourse spaces, different computational linguistic theories about discourses associate diverse non-linguistic structures with the attentional structure. While Grosz and Sidner [GS86] assume that the attentional structure is determined by the intentional structure, which, in turn, usually directly correlates to the subgoal structures inherent to the task domain, Scha and Polani's theory appeals to commonalities among semantic structures of non-overlapping portions of text [SP88].

Apart from this hierarchy of the global focus, the existence of a mechanism of the *local focus* is also generally assumed to reflect the shifting of the focus from one utterance to the next. The task of such a mechanism is primarily to determine the relative value of salience of individual elements in the discourse spaces [Rei85]. The objects in the utterance produced last, being in effect the most highlighted elements in the focused space are called the *focal centers*.

10.2.2 Global Focus

The change of the focus status does not only happen in terms of single derivation steps, that is nodes in a proof tree, but also in terms of larger granularity. Besides the evidence supporting global focus movement in general, its existence is also evidenced by phenomena concerning special subsequent references in argumentative texts [Hua90].

Our segmentation of the proof discourse model directly follows Grosz and Sidner [GS86] by postulating a correlation between the attentional structure and the intentional structure. As outlined previously, the presentation of a proof is generally a combination of top-down planning and bottom-up presentation. While the former requires efforts and concentration, the latter happens usually in a more spontaneous way. As to be expected, the local focus is updated by each single utterance, while there are larger attentional units associated with subgoals created during the planning process. Technically, the intermediate conclusions contributing to the proof of a subgoal constitute an intentional unit, called a *proof unit*. A proof unit is intuitively a subproof upon which a writer once concentrates during this presentation process. Actually, they are special cases of discourse spaces. There is a *one-to-one correlation* between the *subgoal hierarchy* and the *attentional hierarchy*.

Indirectly, there are also correlations between proof units and other intrinsic structures in proofs. The planning process, as will be shown in Chapter 11, is strongly influenced by the *proof-time subgoal structure* and constrained by other proof communicative norms, many of them associated with various structures inherent to the proof under presentation. Consequently, there is also an indirect correlation between the attentional hierarchy and both the proof-time subgoal structure and logical structure associated with the structural ND rules (compare Chapter 4). In other words, a proof unit is usually one of the following:

1. a subproof resulting from the application of those inference methods that are crucial to the success of achieving a particular proof;

2. a subproof consisting of nodes grouped together by shared common assumptions,

3. a subproof rooted at a node derived by a structural natural deduction rule.

Note that the three structures above may overlap.

Dynamically, a unit is created when the corresponding subgoal is posted by the planning procedure, and it usually fades out after the establishment of its proof, with the exception of the root, which will remain eminent in the outer unit. Elements of such a closed unit may be reactivated again, when it is referred to later from the other part of the proof. Adapting the theory of Reichman for our purpose [Rei85], we assume that each unit may have one of the following status:

- a unit is said to be *open*, if the root of the unit is still awaiting to be conveyed.
  - The *active proof* unit is the innermost proof unit containing the local focus. There is exactly one active proof unit at any moment. The root of the active proof unit is called the *current task*.
  - The *controlling proof unit* is the innermost proof unit containing the active unit. Therefore, there is exactly one controlling proof unit at a time, except when the active proof unit is the outermost proof unit.
  - *precontrol* proof units are proof units containing the controlling proof unit.
- *Closed units* are proof units containing only conveyed proof nodes.

The roots of open units are also called the *pending goals*, reflecting the one-to-one correspondence between the goal hierarchy and the attentional hierarchy.

Example 4

Figure 10.1 is an example of a proof discourse model with five proof units. The part of proof still awaiting presentation is omitted, indicated by the dotted links. Suppose that the nodes in proof unit `U2` are all conveyed, and nodes `10`, `11` and `12` are also conveyed. Furthermore, suppose `10` is the last conveyed intermediate conclusion. According to the discussion above, `U4` is the active unit with node `9` as the current task. `U3` and `U1` are the controlling and precontrol unit, respectively. `U2`, clearly, is closed. `U5` is another open unit whose presentation has not started yet.

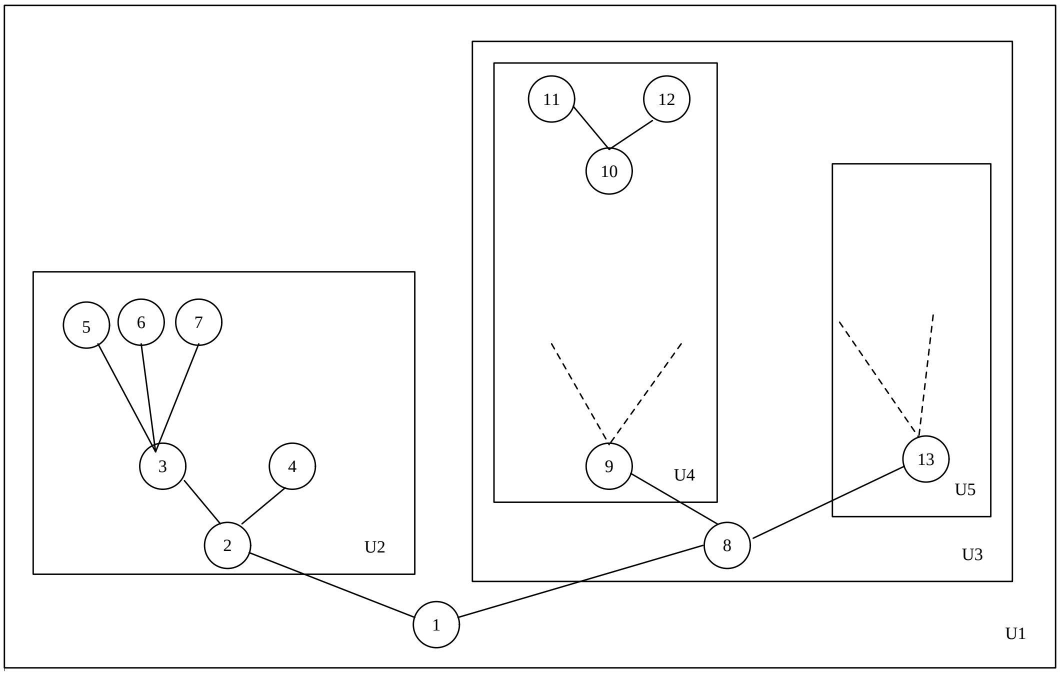


Figure 10.1: An Example of a Proof Discourse Model

10.2.3 Local Focus

At every moment during the presentation of a single proof unit, the intermediate conclusions already conveyed have a differing prominence. The most eminent intermediate conclusion is usually the one just proved, since more complex structures like the theme structure (see section 3.3 in [Lev89]) are not considered in this theory, all semantic elements mentioned in the most eminent intermediate conclusion are the most eminent elements. The most eminent elements in the active discourse space are usually also called the *focal centers* [GJW83]. The only focal centers considered in our theory are the last proved intermediate conclusion, and the semantic objects mentioned in it. In the sequel the last proved intermediate conclusion will be called the *current local focus*, or local focus shortly. In Example 4, if node `10` contains the conclusion which reads "a is an element of S_1", this assertion will be the local focus. The semantic objects "a" and "S_1", together with the assertion itself, are the focal centers.

Apart from the current local focus, a value must be dynamically assigned to all elements in the open proof units, reflecting their relative level of prominence. As will be seen in Chapter 12, these values affect decisions on reference forms to previous conclusions. The focal centers are also used to determine the order of the presentation for subproofs.

10.3 Primitives Updating Proof Discourse History

The presentation history is updated primarily on two occasions: either as the side-effect of the production of an utterance, or when the presentation plan is elaborated. To achieve a more flexible architecture, these updates are not encoded as implicit side-effects of PCAs or planning operators. In contrast, a small set of primitives will be devised to carry out the updates.

Below, we briefly describe several most important primitive operations responsible for the update of PDH:

- ```
  (Set-Conveyed :Conclusion node1
                :Reasons list-of-nodes-1
                :Justification just)
  ```

  marks `node1` as already conveyed and updates its reasons and justification slot correspondingly. Here `just` may be an inference method as described in Part I, or a more informal one such as "Simplify" or "By similarity". In the latter cases, the new reasons usually differ from the original ones in the input proof. If no change needs to be made, the parameter `Reasons` and `Justification` can be omitted.

- `(Open-Unit node)` creates a new proof unit rooted by `node` and makes it the *active* unit. It pushes this active unit on the attentional stack and updates the status of the other existing open units.

- `(Close-Unit node)` marks the attentional unit rooted by `node` as closed, pops the attentional stack and returns to the controlling unit.

- `(Set-Local-Focus node)` marks the node as the current local focus.

10.4 Primitive Proof Communicative Acts

As illustrated in Chapter 9, proof communicative acts are simply speech acts in our domain. As will be discussed in Chapter 12, certain slots in PCAs must undergo reference choices, resulting in preverbal messages (PM).

Although the varieties of utterances appearing in an argumentative texts might be unlimited, at least some major classes of them can be distinguished, from a functional perspective. Currently, thirteen PCAs are identified in this theory. As suggested by the name, proof communicative acts can be specified by the their *natural language realizations* and by the communicative goals they are supposed to fulfill. By natural language realizations, the natural language utterances a corresponding preverbal message brings forth is meant. The communicative goals are specified in terms of the changes of the PDH of the writer, which should be followed by the reader. These goals are only illustrated in an informal way, since no formal planning at the level of PCA is considered.
  ```

In our theory, every PCA has as goal a combination of the following subgoals:

1. Conveying steps of derivations, and thereby convert some unconveyed nodes into conveyed nodes.

2. Guiding the attention of the reader, bringing about an update of the global attentional structure. These PCAs sometimes also convey a partial plan for the further presentation. Effects of this group of PCAs include:

   (a) creating new proof units, setting up partially premises and the goal of the new unit,

   (b) close the current unit,

   (c) reallocate the attention of the reader from one proof unit to another.

The second group of PCAs are called meta-technical utterances (MTU) in [Zuk86]. Discussed within a tutoring context, her MTUs cover a broader type of text than our PCAs do. A model which simulates user's reaction to utterances is employed to motivate the generation of MTUs.

PCAs Conveying a Derivation

There are several PCAs informing the derivation of a new intermediate conclusion from existing ones. `Derive` is the simplest one:

```
(Derive Reasons: list-of-nodes
        Intermediate-Results: list-of-nodes
        Derived-Formula: node
        Method: method)
```

This PCA has a name and four slots:

- `Reasons` is a list of proof nodes (each containing a formula) serving as the premises in this step of derivation,

- `Intermediate-Results` is also a list of nodes, being intermediate results used in the derivation of the `derive-formula`, which are to be mentioned explicitly,

- `Derived-Formula` contains a proof node, being the conclusion of this derivation,

- `Method` contains the inference method used in this derivation, being either an ND inference rule or an assertion (usually a definition or a theorem).

Concretely, suppose the PCA

```
(Derive Reasons: (node1 node2)
        Intermediate-Results: nil
        Derived-Formula: node3
        Method: def-subset)
```

is generated, where the formulas attached to `node1`, `node2` and `node3` are $a \in S_1$, $S_1 \subseteq S_2$ and $a \in S_2$ respectively. Depending on the reference decisions made, different PMs bring forth different verbalizations. The most complete version might be:

> "Because a is an element of S_1 and S_1 is a subset of S_2, according to the definition of subset, a is an element of S_2."

Both slots `Reasons` and `Method` can be omitted, resulting in the verbalization:

> "Therefore, according to the definition of subset, a is an element of S_2."

or

> "Because a is an element of S_1 and S_1 is a subset of S_2, a is an element of S_2."

Note that not only different PMs enforce a variation of the realization, the same PM may also allow for *multiple realizations*. For details, the reader is referred to Appendix B.

Independent of the concrete verbalizations, the following accompanying update activity in the PDH of the writer should be shared by the reader:

```
(Set-Conveyed :Conclusion node3
              :Reasons (node1 node2)
              :Justification def-subset)
(Set-Local-Focus node3)
```

There are four additional PCAs belonging to this group: `Assume`, `Axiom-Instantiation`, `Reformulate`, and `Similar`, see Appendix B.

PCAs Creating New Proof Units

New proof units are created under assorted circumstances, covered by different PCAs. Firstly, a PCA may create new units by setting up new subgoals:

```
(Split-Goal Goal: node,
            Subgoals: list-of-nodes)
```

For instance, if `node` contains the formula $a \in S_2$, and `list-of-nodes` has two elements containing $a \in S_1$ and $S_1 \subset S_2$, respectively, the corresponding PM produces the following natural language utterance:

> "To prove that a is an element of S_2, we will show that a is an element of S_1 and that S_2 is a subset of S_2".

The accompanying update in PDH shared by the reader is performed by the program below:

```
(mapcar #'open-unit list-of-nodes)
```

Secondly, there are PCAs creating new proof units by establishing the hypothesis as well, apart from initiating new subgoals. Two major cases are illustrated below, both directly related to a structural natural deduction rule (see Chapter 4). In connection with the *CASE* rule, the writer may set up two cases after proving $A \vee B$, with A and B as the hypothesis respectively by the following PCA:

(Begin-Cases Goal: *Formula*
Assumptions: $(A\ B)$)

In this case, the verbalization

> "To prove *Formula*, let us consider the two cases by assuming A and B."

will be generated. This PCA is intended to bring about a change in the readers discourse model as illustrated in Figure 10.2, where two new proof units U3 and U4 are created, with the pending goals of showing Formula-1 under the hypothesis A and B, respectively. While normal links represent justification relations, the dotted links indicated not yet presented derivations.

It is interesting to note that the establishment of the pending goals in the PDH of the writer should be more ascribed to the planning process than to the generation of the PCA above. This PCA is intended to bring about this change of PDH on the side of the reader.

A similar situation can be observed when a new pending goal is set up after proving $\exists_x P_x$, with P_a as the new hypothesis. The corresponding PCA is

(`Begin-Assume-Choice-1` x, a)

with the natural language realization:

> "Let a be such an x."

There are four further PCAs in this group: `Begin-Assume-Choice-2`, `Case-First`, `New-lemma`, `Proof-By-Contradiction`.

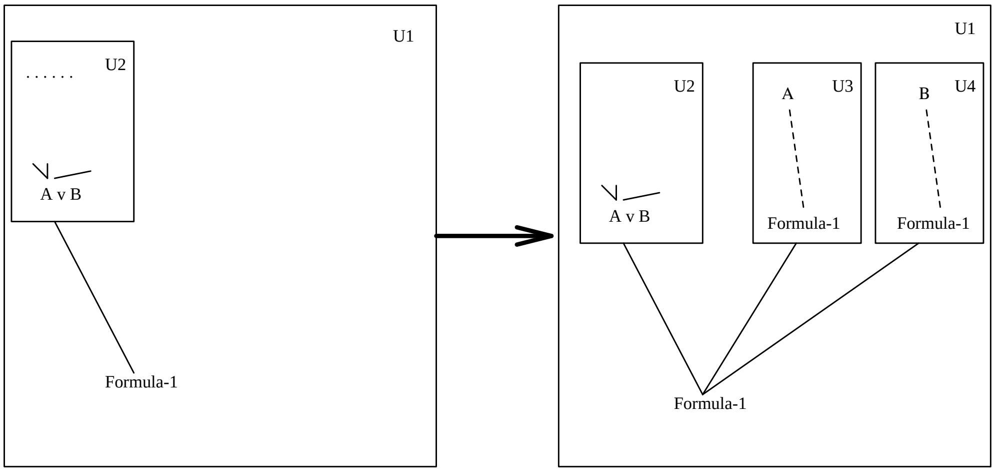


Figure 10.2: PCA-Begin-Case

PCAs Closing Proof Units and Reallocate the Attention

The PCA `Case-Second` first closes the active unit. Then it shifts the attention to a new unit. An example is, after the root of U3 in Figure 10.2 has been conveyed, the PCA

```
(Case-Second)
```

will close U3 and make U4 the active proof unit by generating the sentence:

> "Now let us consider the second case."

Note that, if U4 is not set up as a unit in advance, `Case-Second` augmented with an argument B can be used to create it, with the verbalization below:

> "Now let us consider the second case by assuming B."

There is another PCA in this group, namely `End-Assume-Choice`.

Chapter 11

Generating Proof Communicative Acts

This chapter concentrates on the *macroplanning* process. As sketched out in Chapter 9, the macroplanner has to choose pieces of relative information which are instrumental to the illocutionary goal, and use them to produce a sequence of speech acts. In our case, the illocutionary goal is mainly to convince a reader of the validity of a particular proof. Given this goal, the presentation procedure plans a sequence of PCAs to achieve it.

As briefly sketched out in Section 9.1, most current NL generation systems work under the hypothesis that language generation is planned behavior and therefore adopt a hierarchical planning approach [Hov88, Moo89, Dal92, Rei91]. Nevertheless some psychological theories indicate that language has an unplanned, spontaneous aspect [Och79] as well. Based on this observation, researchers have also exploited organizing text with respect to some local relation. Sibun [Sib90] implemented a system for generating descriptions of objects with a strong domain structure, such as houses, chips and families. Once a discourse is started, local structures suggest the next objects available. Instead of planning globally, short-range strategies are employed to organize a short segment of text.

11.1 A Combination of Top-Down Planning and Bottom-up Presentation

In our model, an attempt is made to integrate the standard *hierarchical planning* and the *unplanned local organization* in a complementary way. Technically, the macroplanning is cast as the combination of *top-down planning* and *focus-guided bottom-up* presentation. In the top-down planning mode, the task of presenting a particular proof is split into subtasks of presenting subproofs, and PCAs are chosen both to present primitive tasks and to mediate between subtasks. While the overall planning mechanism follows the RST based planning approach [Moo89, Rei91], the planning operators more closely resemble the schemata in schema-based planning [McK85]. The way a task is split depends mainly on the global structure of a proof. The recognition of such patterns requires both effort and concentration. The subgoals are therefore also explicitly posted. The bottom-up presentation mode is devised to simulate the unplanned phenomena, where the next node to be presented is picked out under the guidance of the mechanism of the local focus in a more spontaneous way. It is called bottom-up since one new intermediate conclusion is chosen and presented, using already presented intermediate conclusions as its logical premises. The bottom-up presentation procedure takes over the control when no global communicative knowledge is applicable.

The presentation knowledge for both modes is uniformly encoded as *presentation operators*, similar to the plan operators in other generation systems [Hov88, Moo89, Dal92, Rei91]. In general, presentation operators map an original presentation task into a sequence of subtasks and finally into a sequence of PCAs. Clearly, these mappings are not arbitrary, but constrained by *communicative norms*, including general communicative norms and communicative norms specific for argumentative texts, especially for mathematical proofs. Embodying such communicative norms, presentation operators ensure the coherence of the texts produced. Some operators are sensitive to proof-time information, primarily the subgoal hierarchy as well as the order in which these subgoals are created or satisfied. Others work entirely starting from the resulting proof.

The global style of a piece of text produced has to be determined by committing to a global *rhetorical strategy*, since the diversity of the existing proof texts indicates the flexibility of such communicative norms. In other words, the presentation operators represent only the general constraints or strong tendencies motivating to act in a certain way. In many cases, the choice of operators is by no means unique. On the contrary, the choice is strongly influenced by various pragmatic aspects, such as the knowledge level of the addressee, the writer's goal to apply a specific educational strategy, or the goal to emphasize a mathematically interesting part of proof. In other works, such pragmatic aspects of a conversation are first mapped onto a group of rhetorical goals, which are associated with rhetorical strategies, see [Hov90]. The extra level of rhetorical goals is meaningful, because one rhetorical goal can fulfill different interpersonal goals. A simple treatment is adopted in this theory, where each *interpersonal goal* is associated directly with a *rhetorical strategy*, which essentially enforces a partial order on the set of presentation operators in every concrete situation. In other words, in a concrete circumstance the presentation operators are given different priorities by a rhetorical strategy. In Section 11.5, a presentation example will be shown with a very simple strategy which practically enforces a fixed partial order.

Based on the discussion above, the process of macroplanning can be sketched out below. A procedure `Present` always receives the control after initialization. It takes as input a subproof and iterates until its presentation finishes. Initially, `Present` takes the entire proof as input. Each loop tries first to find a suitable top-down presentation operator and executes it. The execution of a top-down planning operator may lead to more than one recursive call to `Present` itself. In other words, the hierarchical planning is realized by recursive calls top-down planning operators. Otherwise, if no top-down operator is applicable, the control is passed on to the procedure `bottom-up-present`, which presents the derivation of one new intermediate conclusion. This process is captured by the program segment below.

```
(defun Present (task); task is a subproof to be presented
  (open-unit task)
  (loop until finished
    (if (top-down planning possible)
        (choose and apply a top-down operator)   ;then
        (choose and apply a bottom-up operator)) ;else
   (close-unit task)))
```

In case more than one applicable operator is found, the most appropriate one is chosen by comparing the global style set by a user (compare Chapter 13), and the stylistic features of operators (compare Section 11.2). Note that the creation of new attentional units is directly associated with the planning of explicitly posted goals. Note also, the treatment of possible failures in applying operators is omitted here. Below, the general structure of presentation operators is described. Section 11.3 and 11.4 then deal with the top-down presentation operators and the focus-guided bottom-up presentation. In Section 11.5, the macroplanning process is illustrated with the help of an example.

11.2 General Structure of Presentation Operators

Before introducing the concrete presentation operators, this section first lays down the general structure for both top-down and bottom-up presentation operators.

The General Structure

All presentation operators have the following four slots:

- The *proof* slot contains a proof schema, which characterizes the syntactical structure of a proof, which this operator is designed for. It plays the role of the goal slot in the traditional planning framework. In the planning process, the metavariables involved in the proof schema will be matched against a concrete subproof being the current task.
- The *applicability Condition* slot contains a predicate, being the applicable condition of this operator.
- The *acts* slot contains a Lisp style program which carries out essentially a sequence of presentation tasks. They are either PCAs, or recursive goals for subproofs.
- The *features*: a list of features are attached to each operator, to help to select the best of a set of applicable operators.

Primitive Predicates for the Applicability Conditions

The following is a list of primitive predicates used to specify applicability conditions.

- `(task-p node)`: is the `node` the root of the subproof which is the current goal?
- `(local-focus-p node)`: is the `node` the local focus?
- `(conveyed-p list-of-nodes)`: is `list-of-nodes` a list of already conveyed nodes?
- `(not-conveyed-p list-of-nodes)`: is `list-of-nodes` a list of still unconveyed nodes?
- `(assertion-p M)`: is the inference method `M` an assertion?
- `(axiom-in-KB-p node)`: is the formula contained in `node` an axiom in the mathematical knowledge base?
- `(in-unit-p unit node)`: is `node` within the attention proof unit `unit`?
- `(in-subproof-p node1 node2)`: is `node2` within the subproof rooted by `node1`?
- `(reformulation-p M)`: is the inference method `M` a reformulation method (defined in Section 11.4.2)?

Features of Operators

A list of features is attached to each operator, to help to single out the best one from the set of applicable operators. Currently one can be chosen from each of the four groups below:

- the features *top-down* and *bottom-up* indicate if the corresponding operator should be used for top-down planning or bottom-up presentation;
- the features *compulsory*, *specific*, and *general* indicate the specificity of the operators with respect to the types of proofs segments they are devised to present;
- the features, *implicitly* and *explicitly* make sense only for top-down operators, reflecting the explicitness the top-down splitting of the proof to be presented is made in the verbalization;
- the features *detailed* and *abstract* reflect if the corresponding operator gives a detailed account of the proof, or only a simplified version.

The last two groups of features are stylistic, indicating the styles of the text the corresponding operators produce. The global style of the text produced has to be determined by committing to a global *rhetorical strategy*, represented as an ordered list of features (compare Chapter 13).

11.3 Top-Down Presentation Operators

This sections elaborates on the communicative norms concerning how goals may be expanded to subgoals, or more precisely, how a proof to be presented may be split into subproofs, as well as how the hierarchically structured subproofs can be mapped onto some linear order for presentation. In contrast with operators employed in RST-based planners which split goals according to the rhetorical structures, our operators encode standard pattern for presenting proofs into schemata, which contain subgoals. This body of knowledge, as we have mentioned above, contains primarily constraints or strong tendencies motivating to act in a specific way. In our model, these constraints and tendencies are encoded in the form of *presentation operators*. The interpersonal goals are influential in choosing among presentation operators, and thereby determine the overall style of the proof. The top-down presentation operators are roughly divided into two categories:

- those containing complex schemata for the presentation of proofs of a specific pattern,
- those embodying general presentation norms, concerning splitting proofs and ordering subgoals.

11.3.1 Proof Structure Operators

This section introduces top-down planning operators devised for proof segments with a to specific structure. The first is associated with proofs containing cases. A corresponding schema of such a proof tree is shown in Figure 11.1.

$$\dfrac{\dfrac{\dfrac{\cdots}{\vdots}}{?Label_4 : \triangle \vdash F \vee G}, \qquad \dfrac{\dfrac{F \vdash F}{\vdots}}{?Label_2 : \triangle, F \vdash Q}, \qquad \dfrac{\dfrac{G \vdash G}{\vdots}}{?Label_3 : \triangle, G \vdash Q}}{?Label_1 : \triangle \vdash Q}\,\mathrm{CASE}$$

Figure 11.1: A Schema Involving Cases

In two circumstances a writer may recognize that he is confronted with a proof segment containing cases. First, when the subproof in 11.1 is the current presentation task, tested by (task $?Label_1$)[1]. Second, when the disjunction $F \vee G$ has just been presented in the bottom-up mode, tested by (local-focus $?Label_4$). No matter in which order the three subtrees are proved at the time the proof was created, there is a communicative norm motivating the writer to first try to present the part leading to $F \vee G$, and then present the proofs of the two cases. Besides this presentation order, this piece of communication norm contains also knowledge about the PCAs used to mediate between parts of proofs. This procedure is exactly captured by the presentation operator below.

[1]Labels stand for the corresponding nodes

Case-Implicit

- Proof: as given in Figure 11.1
- Applicability Condition: ((task ?$Label_1$) $\vee$ (local-focus ?$Label_4$)) $\wedge$ (not-conveyed (?$Label_2$?$Label_3$))
- Acts:

```
(if (not-conveyed ?Label4)
    (present ?Label4))     ;subgoal 1
(Case-First F)
(present ?Label2)          ;subgoal 2
(Case-Second G)
(present ?Label3)          ;subgoal 3
(set-conveyed ?Label1))
```

- features: (top-down compulsory implicit)

The application of this operator posts three new subgoals. The entire verbalization is of the pattern below:

<Verbalization subproof ?$Label_4$ >
"First, let us consider the first case by assuming F."
<Verbalization subproof ?$Label_2$ >
"Next, we consider the second case by assuming G."
<Verbalization subproof ?$Label_3$ >

Note that it has the feature value "implicit" since the splitting into three subgoals is not made explicit. To achieve a certain educational effect, the writer might decide to present the proof in a more explicit way by indicating the splitting explicitly at the beginning. This can be achieved by adding the PCA below to the beginning of the ACTs slot.

```
(Subgoal Label1
         (?Label4 ?Label2 ?Label3))
```

It brings forth the verbalization:

"To prove Q, let us first prove $F \vee G$, and consider the two cases separately."

This explicit version is realized in the presentation operator below.

Case-Explicit

- Proof: as given in Figure 11.1.
- Applicable Condition: ((task $?Label_1$) $\vee$ (local-focus $?Label_4$)) $\wedge$ (not-conveyed ($?Label_2$, $?Label_3$))
- Acts:

```
(cond ((not-conveyed ?Label4)              ;;;first case
       (Subgoal Label1 (?Label4?Label2 ?Label3))
       (Present ?Label4))
      (T (Subgoal ?Label1 (?Label2, ?Label3)))) ;;;second case
(Case-First, P)
(present ?Label2)
(Case-Second, Q)
(present ?Label3)
(Set-Conveyed ?Label1))
```

- Features: (top-down compulsory explicit)

A careful reader might at this point ask why the two operators above are not combined into one, by combining the two programs in the slot "Acts". The answer is that this modularization is determined by our design decision to keep the more *stylistic choices* separated from *general constraints* guaranteeing the minimal coherence of the text produced. While the latter is hardwired into the operators as their applicable conditions, the former is encoded as a list of features. The main aim is to realize stylistic commitments in terms of choices of presentation operators.

The two presentation operators introduced so far encode communicative norms associated with proof nodes justified by the so-called *structural rules* of our ND calculus (compare Chapter 4). There are further ten operators of this group: `Assume-Choice`, `Case-Complementary`, `Choice-Explicit-1`, `Choice-Explicit-2`, `Deduction-Top-Down`, `Derive-Choice-1`, `Indirect-Proof-Explicit`, `Indirect-Proof-Implicit`, `Lemma`.

Communicative norms are also found associated with proofs justified by the application of a specific definition or theorem (collectively called an assertion). In particular, it is usual for a writer to present the derivation of the premises required by a certain assertion in the order in which they are formulated in the assertion itself. For instance, if R is proved as an equivalent relation through its *identity*, *associativity*, and *transitivity* according to the definition of equivalent relation, where these three conditions are specified in the above order, a writer will usually present the subproofs following this order. Below is the implicit version of the corresponding presentation operator.

Assertion-Implicit

- Proof: $\dfrac{?Label_1 : subproof_1, \ldots, ?Label_n : subproof_n}{?Label_{n+1} : A \vdash Q} ?M$
- Applicability Condition: $((\exists_i$ local-focus $?Label_i) \vee$ (task $?Label_{n+1}))$ $\wedge(\exists_i$ (not-conveyed $?Label_i)) \wedge$ (assertion $?M$)
- Acts:

```
(order the ?label_i in an order intrinsic to  ?M)
(go through the above ordered list
    (if (not-conveyed ?Label_i)
        (present Label_i)))
```

- Features: (top-down compulsory implicit detailed)

11.3.2 Ordering and Splitting Operators

The operators discussed so far contain both ordering and presentation constraints. As we have seen, they embody largely communication schemata concerning proofs exhibiting particular logical structures. The presentation operators below, in contrast, perform a simple task according to some general text organization principle, they either

- enforce an order on subproofs in the proof to be presented, or
- split the task of presenting a proof with ordered subproofs into subtasks.

Usually, the invocation of an ordering operator is followed by the invocation of a splitting operator. Below, first a splitting operator is introduced.

Split-Top-Down-Implicit

- Proof: a proof P with $P_1, \ldots, P_n$ as ordered subgoals in the subgoal hierarchy.
- Applicability Condition: (task P)
- Acts:

```
(for i:=1 to n
    (if (not-conveyed P_i)
        (present P_i)))
```

- Features: (top-down general implicit detailed)

The ordering operators to be introduced below embody the following three ordering principles:

- the *minimal load* principle,
- the *local focus* principle,
- the *proof time order* principle.

Let us now examine the ordering operators in turn. First a general ordering strategy called *minimal load principle* is introduced, which is discussed also in a broader context ([Lev89, pp. 143]). This principle predicates that a writer usually presents shorter branches before longer ones. The argument of Levelt is rather simple: When one branch is chosen to be described first, the writer has to have the choice node flagged in memory for return. If he follows the shorter branch first, the duration of the load will be shorter. The corresponding presentation operator is given below:

Order-Minimal-Load

- Proof: a proof P with $P_1, \ldots, P_n$ as unordered subproofs.
- Applicability Condition: (task P)
- Acts: (order the subgoals reverse to the size of the subproofs)
- Features: (top-down general implicit)

The next ordering strategy is the so-called *focus principle*. Roughly, this heuristics works on the basis of two tendencies observable in human proof presentation process. Firstly there is a tendency to use the last derived intermediate conclusion directly for further derivation, before opening another line of deduction. For instance, assume the current task is to present a proof of the form in Figure 11.2.

$$\frac{\dfrac{(6), \quad \dfrac{(9), \quad (10)}{(8)}}{(3)}, \qquad \dfrac{\dfrac{(6), \quad (7)}{(4)}, \quad (5)}{(2)}}{(1)}$$

Figure 11.2: Focused Splitting of Proofs

Assume that subtree rooted by node (2) and (3) are subproofs, node (7) is conveyed and node (6) is the last conveyed node (local focus), the presentation procedure will create two subtask for presenting the two subproofs, and reverse the order of the subproofs. The subproof rooted by node (2) is chosen as the first to be presented, since the local focus can be used to derive node (4) directly, but not node (3), which needs also unconveyed node (8).

The second tendency covered by the focused principle can be stated as follows: After proving a property concerning a particular semantic object, the writer usually tries to derive other properties of the same object, as far as they are useful in the current proof, before turning to properties of other semantic objects. The function `focus-select` will be elaborated upon in section 11.4. It chooses a node maximally overlapping with semantic objects in local focus and introducing minimal novel objects. Recall that the semantic objects mentioned in the local focus are called the focal centers. Built on top of this function, the function `focus-select-unit` accepts a collection of subproofs U and a proof node P, and returns one subproof U_i in U, which should be chosen as the first subproof to be presented. An order for an entire collection of subproofs can be computed by repeated calling this function.

```
(defun focus-select-unit (U: set of proof units, P: proof node)
    (until (P is exclusively in one Ui ∈ U)
          (setq P (focus-select P)))
    (return Ui))
```

Below is the focus based ordering operator itself:

Order-Focus

- Proof: a proof P with $P_1, \ldots, P_n$ as unordered subproofs.
- Applicability Condition: (task P)
- Acts: (order the subproofs by repeatedly applying function `focus-select-unit` starting from the local focus)
- Features: (top-down general)

The *proof-time order* principle says that very often a writer will present subproofs in the same order as they are planned and derived in proof-time. This principle is encoded in our last ordering operator below.

Order-Proof-Time

- Proof: a proof P with $P_1, \ldots, P_n$ as unordered subproofs.
- Applicability Condition: (task P)
- Acts: (order the subproofs according to the proof-time order)

- features: (top-down specific)

Note that the ordering operators as describe above are often simultaneously applicable. Therefore, we have implemented an additional ordering and splitting operator called Proof-Time-Structure-Implicit, which virtually enforces a priority among the ordering principles in the order as listed below (for details see Appendix C):

- the *proof time order* principle,
- the *local focus* principle,
- the *minimal load* principle.

11.4 Bottom-Up Presentation

The *bottom-up presentation* process simulates the unplanned part in proof presentation. Continuing from the last presented intermediate conclusion, it follows the local derivation relation to find a next proof node to be presented. In this sense, it very similar to the local organization techniques used in [Sib90].

The bottom-up presentation differs from the top-down planning process primarily in two ways. In the first place, instead of breaking a presentation task into some subtasks according to some explicit communicative knowledge, the writer simply picks out a next node in a bottom-up manner here, starting from the intermediate conclusions recently presented. Although this in fact divides the presentation task into two subtasks, this is not explicitly planned and thus does not create new proof units, since only a local shift of attention takes place. In the second place, a writer usually does not deliberate on choices in the bottom-up presentation mode, rather, his attention is more guided by the mechanism of *local focus* in an automatic manner to the next node via a local derivation relation. Section 11.4.1 first introduces the bottom-up presentation procedure itself, together with the mechanism of local focus. Then, Section 11.4.2 elaborates on the bottom-up presentation operators.

For implementation reasons, the presentation of primitive proofs (proofs of height one) is handled uniformly as part of bottom-up presentation.

11.4.1 The Local Focus Mechanism and the Bottom-Up Presentation Procedure

The node to be presented next is suggested by the mechanism of *local focus*. Although logically any node having the local focus as one of the premises could be chosen for the next step, usually the one with the greatest semantic overlapping with the *focal centers* is preferred. As mentioned above, focal centers are semantic objects mentioned in the proof node which is the local focus. This is based on the observation that if a mathematician has proved a property concerning a particular semantic object, one will tend to continue to talk about this object, or an object relating to it, before turning to totally new objects. Let us examine the situation when the proof in Figure 11.3 is awaiting presentation.

$$\frac{\dfrac{[1]:\ P(a,b)}{[3]:\ Q(a,b)}, \qquad \dfrac{[1]:\ P(a,b),\ [2]:\ S(c)}{[4]:\ R(b,c)}}{[5]:\ Q(a,b) \wedge R(b,c)}$$

Figure 11.3: Local Focus Guided Presentation

Assume that node [1] is the local focus, the set $\{a, b\}$ is the focal center, [2] is a previously presented node and node [5] is the current task. [3] is chosen as the next node to be presented, since it does not (re)introduce any new semantic element and its overlap with the focal center ($\{a, b\}$) is larger than the overlap of [4] with the focal centers ($\{b\}$). This choice is made by the function `focus-select` sketched out below. It takes as input a node task, and returns a node within the subproof rooted by task which satisfies the following condition:

- It uses the local focus as a direct premise.
- All its premises are already conveyed.
- It introduces least new semantic elements or has the greatest overlapping with the local focus.

```
(Defun focus-select (task)
   (let M be a set containing all nodes under task having only
        conveyed premises (including the local focus)
       (if (empty M)
          (add all nodes under task to M having either only
               conveyed premises, or no premises)
       (choose one element in M introducing least new element or
          having greatest overlapping with the focal center)))
```

This mechanism has a severe restriction, because all semantic objects ($\{a, b\}$ in the example above) which are focal centers are given the same salience value. A refinement is apparently needed.

Now we are prepared to formulate the bottom-up presentation procedure itself. First a node is chosen to be derived in the next step, denoted as the `next-node`. In the situation illustrated in Figure 11.3, node [3] will be `next-node`. Then a bottom-up presentation operator will be chosen and carried out.

```
(defun nl=bu-presentation ()
  (choose a next-node to be presented by calling
    focus-select)
  (choose and apply a bottom-up presentation operator))
```

11.4.2 Bottom-Up Presentation Operators

This section surveys the bottom-up presentation operators. The node suggested by the local focus to be presented next is designated as `next-node`. The most frequently used bottom-up operator is certainly the one which presents one step of the application of an assertion or of a nonstructural rule of natural deduction. It is given below:

Derive-Bottom-Up

- Proof: $\dfrac{?Node_1, \ldots, ?Node_n}{?Node_{n+1}} ?M$
- Applicability Condition: (eq next-node $?Node_{n+1}$) $\wedge$ (conveyed ($?Node_1, \ldots, ?Node_n$))
- Acts:

  (`Derive Derived-Formula:` $?Node_{n+1}$
  `Reasons: (` $?Node_1, \cdots,$ $?Node_n$`)`
  `Method:` $?M$`)))`

- Features: (bottom-up general explicit detailed)

The precondition says, a node $Node_{n+1}$ can be chosen as the next to be presented, if all the predecessors it needs are already conveyed, and it is suggested by the focus mechanism as the next node. The unique PCA generated by this presentation operator is specified in the acts slot.

The question might arise whether this can also be modeled in terms of top-down planning by adding:

- an operator that splits a proof of n nodes into a proof of one node and a proof of (n-1) nodes with the help of local focus, and
- an operator that present a primitive proof containing one node.

It would not only be quite counterintuitive, but also lead to the consequence that every sub-prooftree in the proof discourse would be an attentional unit. This would make the attentional hierarchy useless, and an adequate reference choice for previously presented intermediate conclusions impossible.

A more special bottom-up operator is devised to present the instantiation of an axiom in the database. Since such nodes do not need any predecessors for their derivation, the next-node is used as the unique applicable condition.

Axiom-Instantiation

- Proof: $\dfrac{}{?Node}$Hyp,
- Applicability Condition: (eq next-node $?Node$) $\wedge$ (axiom-in-KB $?Node$)
- Act: (Axiom-Instantiation, Formulas: $?Node$)
- Features: (bottom-up compulsory explicit detailed)

Moreover, there are presentation operators presenting an entire subproof as the next step. Three of them will be discussed below.

If a subproof is considered as *trivial*, the writer may decide to present this subproof as a single derivation, omitting the intermediate nodes in the subproof. Since our theory assumes that bottom-up presentation is guided by the focus mechanism, it is required as the applicable condition that the `next-node` determined by the focus mechanism is in this subtree and the root of the subproof in the active unit. The presentation activity discussed above is captured in the presentation operator Simplify-Bottom-Up.

Simplify-Bottom-Up

- Proof: $\dfrac{\dfrac{?Node_1, \ldots, ?Node_n}{\vdots}}{?Node_{n+1}}$
- Applicability Condition: (in-unit active-unit $?Node_{n+1}$) $\wedge$ (in-subproof next-node $Node_{n+1}$) $\wedge \forall_{i=1\ to\ n}$ (conveyed $?Node_i$) $\wedge$ (simple-proof $?Node_{n+1}$)
- Acts:

  (`Derive Derived-Formula:` $?Node_{n+1}$,
  `Reasons:` $?\ Node_1, \ldots, ?\ Node_n$)
- Features (bottom-up general explicit abstract)

Note that the simplicity of a proof is currently judged by a heuristic function which takes both number of nodes and the type of the derivations in the proof into account. In *PROVERB* currently, a subproof is deemed to be simple if:

- if it does not contain justifications that apply a definition or a theorem in the current theory,

- its size is small enough so that the intermediate steps can be omitted, this size is calculated with respect to the size of the entire proof, and
- its size is large enough so that it can be considered a subproof.

The utterance produced is identical to that produced by the operator **Derive-Bottom-Up**.

Another case where a writer may decide to omit the intermediate nodes in a subproof to be presented is when it is deemed as *similar* to another subproof just undergone presentation. The application condition of this presentation operator resembles that of the operator Simplify-Bottom-Up. The similarity is currently judged by a rather primitive heuristic function, which considers two subproofs as similar, if

- they have identical proof structure with respect to the methods involved, and
- their premises overlap.

Apparently, it has to be refined when more experience is gathered.

Similar-Proof-Bottom-Up

- Proof: $\dfrac{\dfrac{?Node_1\ ,\ldots,\ ?Node_n}{\vdots}}{?Node_{n+1}}$
- Applicability Condition: (in-unit current-unit $?Node_{n+1}$) $\wedge$ (in-subtree next-node $Node_{n+1}$) $\wedge \forall_{i=1\ to\ n}$ (conveyed Label$_i$) $\wedge$ (a subproof considered similar to the one rooted by $?Node_1$ has just been presented.)
- Acts:

  (Similar Derived-Formula: $Node_{n+1}$,
  Reasons: ? $Node_1$,..., ? $Node_n$))
- Features (bottom-up specific explicit abstract)

A sentence of the format "Similarly, since $?Node_1\ ,\ldots,\ ?Node_n$, $Node_{n+1}$." may be generated, although a compound proof is needed to perform this derivation.

Finally, we look at a presentation operator called Reformulation-Implicit-1, aimed at simulating presentation behaviour concerning proof steps usually characterized as syntactic *reformulations*. A classical example is an application of the so-called DeMorgan rule:

$$\frac{\neg(A \vee B)}{\neg A \wedge \neg B}$$

or the propagation of negation through the quantifiers, for example, by applying the rule of inference:

$$\frac{\neg\forall_x F(x)}{\exists_x \neg F(x)}$$

Reformulation steps are usually simple, and therefore are often omitted. Taken for granted that a writer is confronted with the task of presenting a proof of the pattern shown in Figure 11.4, where $?M_2, \ldots, ?M_n$ are reformulating methods. At least three

$$\cfrac{\cfrac{?P_1, \ldots, ?P_m}{\ldots, ?Q_1, \ldots}\,?M_1}{\cfrac{\vdots}{?Q_n}\,?M_n}\,?M_2$$

Figure 11.4: Reformulations

simplified patterns have been identified.

1. if $?Q_n$ is assessed as important in this proof, the writer may decide to mention it explicitly, and omit all reformulating steps. Currently, $?Q_n$ is judged as important if it is used as premise by an application of an assertion, or by a structural ND rule.

2. if there are too many reformulating steps (determined ad hoc in the current implementation), the writer might choose to explicitly mention both $?Q_1$ and $?Q_n$, while omitting the other intermediate reformulating steps.

3. otherwise the writer may explicitly mention $?Q_1$, and omit $?Q_2, ?Q_n$.

These presentation conventions are captured in the presentation operator below.

Reformulation-Implicit-1

- Proof: proof as in Figure 11.4
- Applicability Condition: (in-unit $?Q_n$ current-unit) $\wedge$ (eq next-node $?Q_1$)) $\wedge\forall_{i=1 \ to \ m}$ (conveyed $?P_m$)$\wedge\forall_{i=1 \ to \ n}$ (reformulation $?M_i$)
- ACTS:

```
(cond ((important ?Qn)      ;;;case 1
       (Derive Derived-Formula: ?Qn
               Reasons: ?P1, ..., ?Pm
               Method: ?M1))
       ((> n 4)              ;;;case 2
        (Derive Derived-Formula: ?Q1
                      Reasons: ?P1, ..., ?Pm
                      Method: ?M1)
        (Reformulate Derived-Formula: ?Qn))
```

```
(T (Derive Derived-Formula: ?Q1 ;;;case 3
           Reasons: ?P1, ···, ?Pm
           Method: ?M1))
```

- features (bottom-up compulsory explicit abstract)

Before closing this section, we want to point out two important properties of the collection of presentation operators introduced up to now. First, it is *complete* in the sense that for every proof, there is always a sequence of applicable operators which accomplishes the task of presenting the given proof. Secondly, it is *indeterministic* in the sense that there is normally more than one such sequence of applicable operators. A writer may however single out one sequence of operators by adhering to a particular style appropriate for his purpose.

11.5 A Presentation Example

In this section, the entire macroplanning process is illustrated with the help of an example. This will be done by taking several snapshots of the presentation process as it is implemented in the *presentation* module of the system *PROVERB*. The input of the presentation module is an ND style proof at the assertion level, abstracted from a machine-generated ND proof, to be described in Part III. The assertion level proof to be presented given below is represented in a linearized version of ND proofs, introduced in [And80]. In this formalism, every proof is a sequence of proof lines, each of them is of the form:

$$\textit{Label} \qquad \triangle \quad \vdash \quad \textit{Derived-Formula} \qquad\qquad \textit{Justification}(\textit{reason-pointers})$$

where *Justification* is either a rule of inference in $\mathcal{NK}$, or an assertion, which justifies the derivation of the *Derived-Formula* using formulas in lines pointed to by *reason-pointers* as the premises. $\triangle$ is a finite set of formulas, hypotheses upon which the derived formula depends, and it can be ignored for our purpose. The proof to be presented is a machine generated proof for a theorem taken from a textbook about semigroup and automata [Deu71].

Satz 1.9 in [Deu71]:

Let F be a group and U a subgroup of F, if 1 and 1_U are unit elements of F and U respectively, then $1 = 1_U$.

The definitions of *semigroup*, *group*, and *unit* are obvious. $solution(a, b, c, F, *)$ should be read as "c is a solution of the equation $a * x = b$ in F."

Abstracted Proof about Unit Element of Subgroups

NNo	S;D		Formula	Reason
			Definitions	
1.	;1	$\vdash$	$\forall_{U,F}\ U \subset F \Leftrightarrow \forall_x\ x \in U \Rightarrow x \in F$	(Def-Subset)
2.	;2	$\vdash$	$\forall_{1,F,*}\ unit(F,1,*) \Leftrightarrow semigroup(F,*) \wedge 1 \in F \wedge (\forall_f\ f \in F \Rightarrow 1 * f = f * 1 = f)$	(Def-Semigroup*Unit)
3.	;3	$\vdash$	$\forall_{F,*}\ group(F,*) \Leftrightarrow semigroup(F,*) \wedge (\exists_1\ unit(F,1,*) \wedge (\forall_f\ f \in F \Rightarrow \exists_{f^{-1}}\ f^{-1} \in F \wedge f^{-1} * f = 1))$	(Def-Group)
4.	;4	$\vdash$	$\forall_{U,F,*}\ subgroup(U,F,*) \Leftrightarrow semigroup(F,*) \wedge U \subset F \wedge group(U,*)$	(Def-Subgroup)
5.	;5	$\vdash$	$\forall_{f,g,x,F,*}\ solution(f,g,x,F,*) \Leftrightarrow semigroup(F,*) \wedge f,g,x \in F \wedge f * x = g$	(Def-Solution)
6.	;6	$\vdash$	$\forall_{f,g,x,y,F,*}\ group(F,*) \wedge solution(f,g,x,F,*) \wedge solution(f,g,y,F,*) \Rightarrow x = y$	(Th-solution)
			The Proof	
7.	;7	$\vdash$	$group(F,*) \wedge subgroup(U,F,*) \wedge unit(F,1,*) \wedge unit(U,1_U,*)$	(Hyp)
8.	4;7	$\vdash$	$U \subset F$	(Def-Subgroup 7)
9.	2;7	$\vdash$	$1_U \in U$	(Def-Semigroup*Unit 7)
10.	2;7	$\vdash$	$\exists_x\ x \in U$	($\exists$ 9)
11.	;11	$\vdash$	$u_1 \in U$	(Hyp)
12.	2;7,11	$\vdash$	$u_1 * 1_U = u_1$	(Def-Semigroup*Unit 7 11)
13.	1,4;7,11	$\vdash$	$u_1 \in F$	(Def-Subset 8 11)
14.	1,2,4;7	$\vdash$	$1_U \in F$	(Def-Subset 8 9)
15.	3;7	$\vdash$	$semigroup(F,*)$	(Def-Group 7)
16.	1,2,3,4,5;7,11	$\vdash$	$solution(u_1,u_1,1_U,F,*)$	(Def-Solution 15 13 14 12)
17.	1,2,4;7,11	$\vdash$	$u_1 * 1 = u_1$	(Def-Semigroup*Unit 7 13)
18.	2;7	$\vdash$	$1 \in F$	(Def-Semigroup*Unit 7)
19.	1,2,3,4;7,11	$\vdash$	$solution(u_1,u_1,1,F,*)$	(Def-Solution 15 13 18 17)
20.	6,1,2,3,4,5;7,11	$\vdash$	$1 = 1_U$	(Th-Solution 7 16 19)
21.	1,2,3,4,5,6;7	$\vdash$	$1 = 1_U$	(Choice 10 20)
22.	1,2,3,4,5,6;	$\vdash$	$group(F,*) \wedge subgroup(U,F,*) \wedge unit(F,1,*) \wedge unit(U,1_U,*) \Rightarrow 1 = 1_U$	(Ded 7 21)

The corresponding proof tree is shown in Figure 11.5[2]. Children of nodes are given in the order as they will be presented. The circles denote nodes which are first derived at this place, and nodes in the form of small boxes are copies of some previously derived nodes, which are used as premises again. The big boxes represent the attentional units called *proof units*, created during the presentation process. Apparently, the proof units form a hierarchy. The output proof in English is repeated in Figure 11.6.

[2]Multiple copies of proof lines in the linearized version are needed to construct a proof tree. We will continue to used this standard notion, although it is technically a DAG (directed acyclic graph).

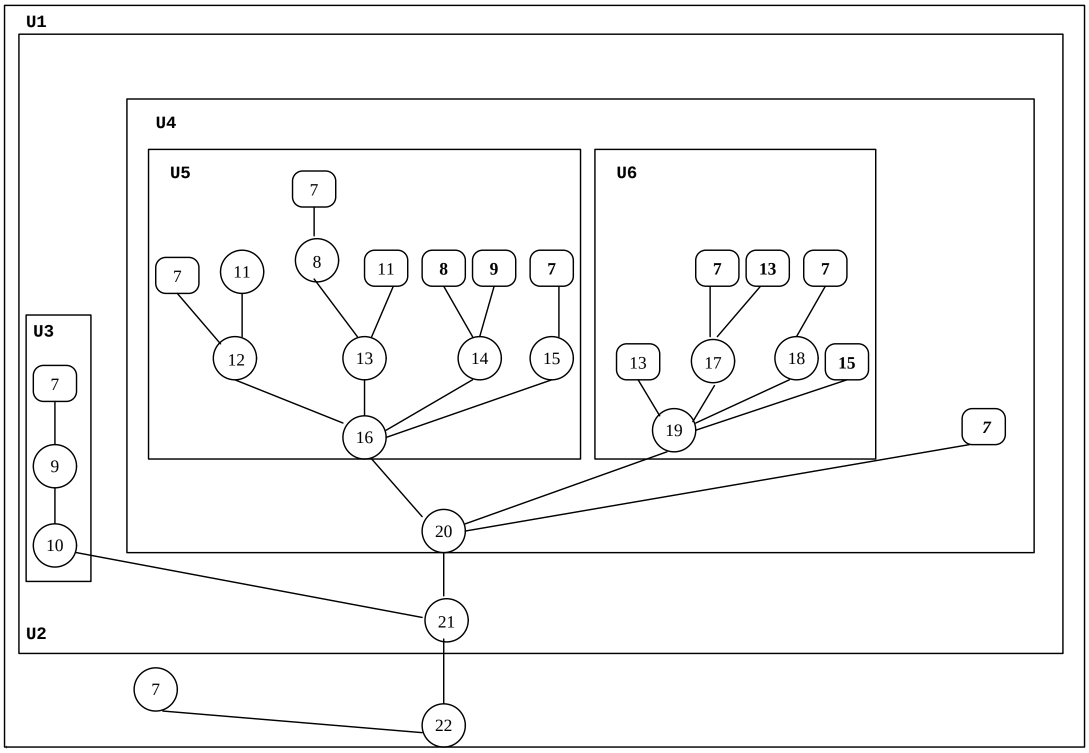


Figure 11.5: Proof Tree for Satz 1.9

The Natural Language Proof

(1) Let F be a group and U be a subgroup of F and 1 be a unit element of F and 1_U be a unit element of U. (2) According to the definition of unit element $1_U \in U$. (3) Therefore there is an X, $X \in U$. (4) Now suppose that u_1 is such an X. (5) According to the definition of unit element $u_1 * 1_U = u_1$. (6) Since U is a subgroup of F $U \subset F$. (7) Therefore $u_1 \in F$. (8) Similarly $1_U \in F$ since $1_U \in U$. (9) Since F is a group F is a semigroup. (10) Since $u_1 * 1_U = u_1$ 1_U is a solution of the equation $u_1 * X = u_1$. (11) Since 1 is a unit element of F $u_1 * 1 = u_1$. (12) Since 1 is a unit element of F $1 \in F$. (13) Since $u_1 \in F$ 1 is a solution of the equation $u_1 * X = u_1$. (14) Since F is a group $1_U = 1$ by the uniqueness of solution. (15) This conclusion is independent of the choice of the element u_1.

Figure 11.6: A NL Proof Generated by *PROVERB*

As will be discussed in Chapter 13, heuristics are employed to simulate subtrees in the input proof as possible conceptual subproofs during proof time. In this example, the system also commits itself to a standard rhetorical strategy specified as an ordered list of features. The information useful for our purpose after initialization is listed below (the global style of the text can be influenced by permuting the list of styles, compare Section 9.3):

Initialization

- Proof-time-subproofs: (10,16,19,20,21,22)
- Task-Stack:(22)
- Style: (compulsory specific general implicit explicit abstract detailed)

The text planning process is illustrated by examining the sequence of plan operators invoked as well as the actions they take. Actually, in connection with every execution of presentation operators, the following items are shown:

- OP: records the name of the presentation operator invoked.
- PCA, PM, and NL: there can be more than one group of these three items, recording the proof communicative acts decided upon by the current presentation operator, the corresponding preverbal messages produced in the microplanning stage (to be discussed in Section 12.5), as well as the final wording in natural language. These actions are carried out immediately after the current presentation operator is invoked. If a PM item is identical to the corresponding PCA item, the former is omitted. Other PCAs must be incorporated into units around other presentation operators (see the items Pre and Post).
- Nodes-Conveyed: a list of proof nodes conveyed with the execution of this presentation operator.
- Pre and Post: due to the recursive nature of the presentation process, however, some of the actions decided upon by one operator have to be put into items of other operators. These two items are devised to accommodate such actions, as well as other bookkeeping actions.
- Task-Stack: it is maintained for the convenience of the discussion, although this task is carried out by the recursion handling mechanism of Common Lisp.

Normally, finished proofs end with a proof line justified by the *DEDuction* rule of $\mathcal{NK}$, discharging the hypothesis by constructing an implication as the final theorem (compare Chapter 4). If the proof is achieved as a proof by contradiction, the last line will be justified instead by one of the rules for indirect proof. Therefore, normally the top-down planning operator **Deduction** is applied at the very beginning, which establishes first the assumption for the entire proof (line 7). This is recorded as the first step below.

Step 1

- OP: Deduction, top-down

- PCA:

```
(Derive Derived-Formula:  group(F,*) ∧ subgroup(U,F,*)
                          ∧ unit(F,1,*) ∧ unit(U,1_U,*)
        Method: Hyp)
```

- NL: "Let F be a group and U a subgroup of F and 1 be a unit element of F and 1_U a unit element of U."
- Nodes-Conveyed: 7
- Task-stack: (**21**)

As the second Step , the planner finds there are two top-down applicable presentation operators: **Choice-Explicit-2** and its implicit counterpart **Choice-Implicit-2**. Since the planer has committed to an explicit rhetorical strategy, it chooses the explicit one. **Choice-Explicit-2** does currently nothing more than pushing two subgoals into the presentation stack, namely, node 10 and 20. This recorded as the second step:

Step 2:

- OP: Choice-ExpliCit-2, top-down
- Task-stack: (**10**,20,21)

Now the current task is the presentation of subproof rooted by node 10, and there is no applicable top-down operators. At this moment, the system enters the bottom-up mode, choosing the next node to be presented guided by the local focus mechanism. It is node 9 that the function `focus-select` suggests as the next node. In this case, it is fairly trivial since node 9 is both the only one depending on the local focus (node 7), which the current task (node 10) depends on, see Section 11.4. This bottom-up presentation step is recorded in Step 3. A more complicated case for choosing the next node guided by the local focus can be found in Step 7, see Appendix A.

Step 3:

- Pre: (open-unit 10)(next-node 9)
- OP: Derive-Bottom-Up
- PCA:

```
(Derive Derived-Formula:  1_U ∈U
        Reasons:  unit(1_U,U,*)
        Method: Def-Semigroup*Unit)
```

- PM:

```
(Derive Derived-Formula:  1_U ∈U
        Method: Def-Semigroup*Unit)
```

- NL: "According to the definition of unit element, $1_U \in U$."
- Nodes-Conveyed: 9
- Task-stack: (**10**, 20, 21)

Let us now jump over one bottom-up presentation step and consider Step 5. Up to now, sentence 1 to 3 in Figure 9.7 are already uttered, the current task is node 20. At this point, no communicative knowledge specific to the proof structure is at the disposal of the planner. Since the subproofs rooted at nodes 16 and 19 are initialized as subgoals at the proof time, nevertheless, and since the presentation process is accompanied by a verification process, the planner may decide to present the two subproofs exactly according to the order in which they are proved. In our system, however, since normally no information about the proof time order is available, a focus based heuristic is used to single out one unit to undergo the presentation process first. This heuristics is similar to that used in bottom-up presentation in section 11.3. In this case, the unit rooted at node 16 is chosen as the first because the function `Focus-Select` suggests that node 12 should be presented next, and node 12 belongs to the unit rooted at node 16 exclusively. Note that purely logically both node 12 and node 8 could be picked out as the next node. But the function `focus-select` chooses 12 because it states $u_1 * 1_U = u_1$, a property of the semantic object u_1 introduced in sentence 4, while node 8 states $U \subset F$, having nothing to do with u_1. Although this operator does not produce any PCA, it adds two subgoals to the task stack, namely, node 16 and 19.

Step 5:

- Pre:
  - PCA: (Begin-Assume-Choice-1, u_1, x), OP: 2
  - NL: "Now suppose that u_1 is such an x."
  - (node-closed 11)
- OP: Proof-Time-Structure-Implicit, top-down
- Task-stack: (**16**,19,20,21)

Step 6 is again a bottom-up step guided by the focus mechanism, which conveys the node 12.

Step 6:

- Pre: (open-unit 16)(next-node 12)
- OP: Derive, bottom-up
- PCA:

```
(Derive Derived-Formula:  u1 * 1U = u1
        Reasons: (unit(1U, U, *),  u1 ∈U)
        Method: Def-Semigroup*Unit)
```

- PM:

```
(Derive Derived-Formula:  u1 * 1U = u1
        Method: Def-Semigroup*Unit)
```

- NL: "According to the definition of unit element, $u_1 * 1_U = u_1$."
- Nodes-Conveyed: 12
- Task-stack: (**16**,19,20,21)

This example will be continued in the next chapter, where choices of reference forms will be determined in the PCAs to produce preverbal messages.

Chapter 12

Reference Choices in PCAs

In most NL generation systems, the more global macroplanning is followed by a micro-planning phase responsible for paragraph and sentence level organization. Since it is out of the scope of this work, this stage of planning is restricted to the treatment of the *reference choices* in PCA. Since most mathematical proofs can be organized in one paragraph, and since in most cases the PCAs employed can be realized in a single sentence, the quality of the text generated will not be significantly affected.

By reference choice the explicitness of the verbalization of certain entities in some PCAs is meant. Concretely, such decisions must be made for both intermediate conclusions used as premises, as well as the inference method in the PCA `Derive`, `reformulate`, `similar-assume-derive`, and `similar-derive`. Since the effect is very similar, the discussion in this chapter will be mostly based on the PCA `Derive`.

As an example, let us look at a PCA generated in the example discussed in Section 11.5:

(`Derive Derived-Formula:` $u * 1_U = u$
`Reasons:` $(unit(1_U, U, *),\ \ u \in U)$
`Method: Def-Semigroup*unit`)

Figure 12.1: An Example PCA

Here, *Derived-Formula* is filled by a new intermediate conclusion the current PCA aims to convey, which is derivable by applying the filler of *Method*, with the filler of *Reasons* as premises. While the new intermediate conclusion to be conveyed will usually be handed over unchanged to the linguistic component, there are alternative choices for referring to both the *Reasons* and the inference *Method*. Depending on the discourse history, the following are some of the possible verbalizations:

Verbalization 1 (inference method omitted):

"Since 1_u is the unit element of U, and u is an element of U, $u * 1_U = u$."

Verbalization 2 (reasons omitted):

> "According to the definition of unit element, $u * 1_U = u$."

As will be illustrated below, the reference choices faced here have much in common with the decisions on *referring expressions* in more general frameworks [Rei85, GS86, Dal88]. It must be pointed out nevertheless, that in our case it is not restricted to noun phrases, as the term referring expression is linguistically defined. As shown in the example above, what in concern can also be a subordinate clause.

Usually, the writer will try to sail a middle course while making reference choices, avoiding neither to be *over-informative* nor *under-informative*. In other words, a co-operative writer will choose a referring expression informative enough for an addressee to deduce the object referred to. This requirement is first stated in the *cooperative principle* of Grice [Gri75a, Lev89]. In the following sections, we will first provide a classification of the referring expressions both for reasons and methods, and then discuss mechanisms for making corresponding decisions.

12.1 A Classification of Reference Forms

This section presents a classification of the possible forms with which mathematicians refer to intermediate conclusions previously proved (called reasons) or to methods of inference. The classification is based on the careful analysis of proofs presented in mathematical textbooks.

12.1.1 Reference Forms for Reasons

The following three reference forms *reasons* have been identified in naturally occurring proofs:

1. The *omit* form: where a reason is not mentioned at all.

   For instance, if the omit form is adopted for the first reason in the PCA given in Figure 12.1, the sentence may be produced:

   > "Since u is an element in U, $u * 1_U = u$."

2. The *explicit* form: where a reason is literally repeated.

   If this form is adopted for both reasons in the PCA given in Figure 12.1, it may yield the sentence:

   > "Since u is an element in U and 1_U is the unit element of U, $u * 1_U = u$."

3. The *implicit* form: By an implicit form it is meant that although nothing is said directly as to a reason, an implicit hint to the reason is nevertheless given in other components of the PCA. That is, in the verbalization of either the inference rule, or the conclusion.

   For example, in the verbalization below the first reason of the PCA in Figure 12.1, "since 1_U is the unit element of U" is hinted at by the inference method which reads "by the definition of unit":

   > "Since u is an element in U, $u * 1_U = u$ by the definition of unit."

Note that although an identical surface structure will be chosen for omit and implicit forms, the existence of an implicit hint in the other part of the verbalization affects understanding of a reader.

12.1.2 Reference Forms for Methods

Besides choosing referring expressions for reasons, the microplanner must select referring expressions for the method of inference in PCAs. Below are the three reference forms identified, which are analogous to the corresponding cases for reasons:

1. the *explicit* form: this is the case where a writer may decide to indicate explicitly which inference rule he is using.

   For instance, explicit translations of domain-specific rules could look like:

   > "by the definition of unit element", or "by the uniqueness of solution."

   Structural ND rules have usually standard verbalizations (compare appendix B).

2. the *omit* form: in this case a word such as "thus" or "therefore" will be used.

3. The *implicit* form: Similar to the implicit form for reasons, an implicit hint to a domain-specific inference method can be given in the verbalization of either the reasons, or the conclusion.

12.2 Making Reference Choices for Reasons

Because reasons are intermediate conclusions proved previously in context, the problem of choosing reference forms for reasons in an argumentative discourse has much in common with the problem of choosing *anaphoric* referring expressions in natural language generation in general. A number of theories have been put forward to account for the phenomenon of pronominalization, which is usually ascribed to the focus mechanism. For this purpose, concepts like activatedness, foregroundness and consciousness are introduced [Kan77, Cha75, Cha76, Kun77]. More recently, the shift of focus has been further investigated in the light of a more structured flow of discourse [Gro77, Sid79, Rei85, GS86, Dal88, Web89]. The issue of salience is also studied in a broader framework in [Pat93, PC93]. Apart from salience, it is also shown that referring expressions are strongly influenced by other aspects of human preference. For example, easily perceivable attributes and basic-level attributes values are preferred [DH91, Dal92, RD92].

Common to all discourse based theories, the update of the focus status is tightly coupled to the *factoring* of the flux of text into segments. As discussed in Chapter 10, different approaches have been suggested to answer the question as to where the segment boundaries lie. We basically follow the approach of Grosz and Sidner [GS86] in that a direct correspondence between the plan hierarchy and the attentional spaces is assumed, since it best reflects the dynamic flow of attention. In this way, the discourse is very naturally segmented into attentional spaces (called *proof unit*), where each space is initiated with the application of a top-down planning operator introduced in section 11.3. Because the top-down planning is supplemented by an unplanned bottom-up presentation, the elementary proof units contains usually a sequence of simple derivations.

With the segmentation problem settled, the decisions on referring expressions in our theory largely follows the approach of Reichman. Reichman handles the reference problem in a more general framework of her discourse grammar (see [Rei85, Chapter 5]). Based on empirical data, Reichman argues that the choice of referring expressions is constrained both by the *status of the discourse space* and by the object's *level of focus* within this space. In her theory, there are seven status assignments a discourse space may have at any given time. Within a discourse space, four levels of focus can be assigned to individual objects: *high*, *medium*, *low*, or *zero*, since there are four major ways of referring to an object using English: by using a pronoun, by name, by a description, or implicitly. Thirteen rules are formulated to assign level of focus to objects when they are activated, either with the initialization of a discourse unit or when they are added to the active unit. Four further rules reassign the level of focus on reentrance of a suspended discourse space. Based on the status assignment of discourse spaces, as well as the level of focus of individual objects, four rules are formulated constraining adequate referring expressions.

In short, Reichman takes into account both the foreground and background status of proof spaces and the level of focus of individual intermediate conclusion. As a simplification for argumentative discourse, the notions of *structural distance* and *textual distance* are introduced.

The *structural distance* of a reason reflects the foreground and background character of the innermost proof unit containing it. Intuitively, reasons that may still remain in the focus of attention at the current point from the structural perspective will be considered as *structurally close*. Otherwise they will be considered as *structurally distant*. If a reason, for instance, is last mentioned or proved in the active proof unit, which is the proof unit a human writer is currently working on, it is more likely that this reason should still remain in his focus of attention. On the other hand, if the reason is in a closed unit and is not the root, it is very likely that the reason has already been moved out of the writer's focus of attention. It is found that the proof structure has a strong influence on the reference forms. Although the notion of foreground and backgroundness might actually be a continuum, our theory only distinguishes between reasons residing in proof units which are *structurally close* or *structurally distant.* Rules assigning this *structural distance* are given in Figure 12.2.

Assignment of Contextual Status for Reasons

1. Reasons in the *active* proof unit are structurally close.
2. Reasons in the *controlling* proof unit are structurally close.
3. Reasons in *closed* units:
   (a) reasons that are root nodes of closed proof units immediate subordinate to the active unit are structurally close.
   (b) Other reasons in closed proof units are structurally distant.
4. Reasons in *precontrol* proof units are structurally distant.

Figure 12.2: Contextual Status for Reasons

Note that the rules are specified with respect to the innermost proof unit containing a proof node. Rule 3 means that only the conclusions of closed subordinated subproofs still remain in the focus of attention. In addition, as a special treatment, premises of the theorem will be defined as both structurally and textually distant, if they are not repeated at the beginning of the proof.

The *textual distance* is used as an approximation to the level of focus of the reason. In general, the level of focus of an object is established when it is activated, and decreases with the flow of discourse. In Reichman's theory, although four levels of focus can be established upon activation, only one is used in the formulation of the four reference rules. In other words, it suffices to track the status *high* alone. Based on the discussion above, only two values are employed to denote the level of focus of individual intermediate conclusions, depending solely on the *textual distance* between the last mentioning of a reason and the current sentence where the reason is referenced to. There is another practical reason supporting a two-valued solution: although three ways of reference for reasons are identified, it is still unclear whether the implicit form and the omit form (compare Section 12.1) actually establish different levels of focus. Currently, we assume they do not. On account of the discussion above, the two levels of focus are called *textually close* and *textually distant*.

In summary, we assume that each intermediate conclusion is put into high focus when it is presented as a newly derived result or cited as a reason supporting the derivation of a further intermediate result. This level of focus decreases, either when a proof unit is moved out of the foreground of discussion, or with the increase of textual distance. On account of the above, the four *reference rules* used in our computational model are given in Figure 12.3.

Choices for Referring Expressions for Reasons

1. If a reason is both structurally and textually close, it will be omitted. As a special treatment for the hypothesis of the entire problem, the omit form will also be chosen for nodes structurally close but textually distant, which have still not been used as a reason.
2. If a reason is structurally close but textually distant, first try to find an implicit form, if not possible, use an explicit form.
3. If a reason is structurally distant but textually close, first try to find an implicit form, if not possible, omit it.
4. An explicit form will be used for reasons that are both structurally and textually distant.

Figure 12.3: Reference Rules for Reasons

Note that the result of applying rule 2 and rule 3 depends on the fact that an implicit form is available, which often interacts with the verbalization of the rest of the PCA. In particular, it interacts with the reference choices for inference methods. In *PROVERB* as it currently stands, this is realized by associating a new word with the verbalization of every predicate, function, and assertion. For instance, suppose the verbalization of $unit(F, 1, *)$ as a reason is "since 1 is an unit element of F", and the verbalization of the definition of unit element as an inference method is "by the definition of the unit element". Both the predicate *unit* and the definition are associated with the same keyword "unit". Therefore, *PROVERB* assumes that the verbalization of the reason $unit(F, 1, *)$ and the verbalization of the definition of unit element hint at each other. Distance is currently calculated in an ad hoc way by counting the PCAs uttered after the corresponding reason was last explicitly mentioned.

12.3 Making Reference Choices for Inference Methods

Like the reference to a reason, the explicitness or implicitness of referring to an inference method at a particular point depends on whether the particular method can be easily called into the foreground of the focus of attention. In contrast to references to reasons, this is evidently irrelevant to the particular discourse structure concerned. Actually it is less concerned with the proof context than with the user's familiarity with the particular inference method. This referring behavior remains the same throughout a whole discourse, similar to the referring behavior relating to the so-called canonical salience [Pat93]. In the case of applications of definitions or theorems, it depends on the reader's familiarity with the corresponding definition or theorem. This is found to be sensitive to the global hierarchy of the mathematical theories [Ker91, SK93]. As it currently stands, *PROVERB* distinguishes only between assertions in the *underlying theories* and assertions belonging to the *current theory*. The reference choice rules for inference methods incorporated in our computational model are listed in Figure 12.4.

Choices for Referring Expressions for Methods

1. Reference Choices for *ND Inference Rules*

   (a) All non-structural ND rules will be omitted (In the case of PCA `DERIVE`, a word like "thus", "hence", etc. will be used), because the readers are supposed to be familiar with the elementary logic.

   (b) All structural ND rule will be explicitly given. Although they are also familiar to the readers, they provide land marks for the overall proof structure.

2. Reference Choices for *Applications of Assertions*

   Readers are assumed to be familiar with definitions and theorems of the "underlying theories" upon which our current theory is based. For example, when we are reasoning about properties of group theory we assume that the users are familiar with basic set theory and the omit form should be chosen for the application of definitions or theorems of basic set theory:

   (a) Applications of definitions and theorems of *underlying theories* will be omitted.

   (b) For applications of definitions or theorems of the *current theory*, try first to find an implicit form. If impossible, an explicit indication will be given.

Figure 12.4: Reference Rules for Methods

12.4 An Integrated Algorithm for Reference Choices

As illustrated above, reference choices for reasons and for methods *interact* with each other. This section describes an algorithm combining the reference choice rules for reason and the reference choice rules for methods, to produce preverbal messages from PCAs. As such, the main task is to utilize the interaction between the two sets of reference rules to eliminate the indeterminacy in both of the rule sets. The indeterminacy lies in rule 1 and 2 in Figure 12.3 and in rule 2(b) in Figure 12.4, which need information on decisions made by the other set of rules. In other words, decisions in one rule set may help to narrow the alternatives in the other set. In *PROVERB* as it currently stands, the reference choice for the inference method is first made. While doing so, *PROVERB* looks ahead and takes the possible reference choices for reasons into account. If still no unique choice can be made, the decision is made according to a predetermined order. Concretely, the explicit form will be chosen for rule 2(b) in Figure 12.4. Below is the current algorithm.

```
(defun nl=reference-form-method
    ;;;determine the reference forms for
    the inference method in the current PCA
    (cond ((method is structural ND rule)   ;;;first case
           (choose "explicit"))
    ((method is non-structure ND rule)      ;;;second case
     (choose "omit"))
    ((method applies a definition or a theorem in underlying
        theories)                           ;;;third case
     (choose "omit"))
    ((method applies a definition or a theorem in the
                                 theory);;;fourth case
     ((if (the method is hinted at by the conclusion or one of the
           explicit reasons) ;;;look ahead for choices for reasons
          (choose "implicit")
          (choose "explicit"))))))

(defun nl=reference-forms ()
  ;;;determine the reference forms for reasons
  ;;;and the method in the current PCA
  (nl=reference-form-method)
  (nl=reference-form-reasons))
```

Note that, after the reference form of the method is already determined, a unique reference form can always be singled out for the reasons. This algorithm handles successfully the special kind of references addressed in this chapter. It should be extended to handle other referring expressions more widely discussed, such as subsequent references to semantic objects, sentential anaphora and event anaphora.

12.5 A Presentation Example (Continued)

Now we continue with the subgroup example introduced in Section 11.5 and illustrate how *PROVERB* decides on reference forms and thereby generates preverbal messages from PCAs.

Let us examine the generation of preverbal messages in the Step 3 (pp. 107) and Step 6 (pp. 109) listed in Section 11.5. The PCA in Step 3 aims to convey that derivation of proof node 9 ($1_U \in U$) from a part of node 7 ($unit(1_U, U, *)$) can be justified by the application of the definition of the unit element in semigroups. The current unit is U3. U2 and U1 are the controlling and precontrol unit, respectively, see Figure 11.5. Since node 7 is in the controlling unit and is mentioned last only two steps previously, it is therefore judged as both structurally and textually close. Rule 1 in Figure 12.3 matches and the *omit* form is chosen. Since the definition of the unit element resides in the current theory as mentioned in the initialization, Rule 2(b) in Figure 12.4 suggests the application of the definition be referred to either implicitly or explicitly. Because the implicit option is ruled out by the omit form for reasons, the explicit form is chosen. Therefore, from the PCA

```
(Derive Derived-Formula: 1_U ∈U
        Reasons: unit(1_U, U, *)
        Method: Def-Semigroup*Unit)
```

the PM below is generated:

```
(Derive Derived-Formula: 1_U ∈U
        Method: Def-Semigroup*Unit)
```

Now we jump to Step 6. Here, a PCA is generated to convey that node 12 ($u_1 * 1_U = u_1$) can be derived from part of node 7 ($unit(1_U, U, *)$) and node 11 ($u_1 \in U$) by applying the definition of the unit element. U5 is now the current unit, with U4 and U2 as controlling and precontrol units. U3 is the unique closed unit. Reason node 7 is now structurally distant but still texually close, and node 11 is in the current unit and is the node last conveyed, therefore, both 7 and 11 are omitted. The reference form for the definition of unit element is decided upon as above. In short, from the PCA

```
(Derive Derived-Formula: u_1 * 1_U = u_1
        Reasons: (unit(1_U, U, *),  u_1 ∈U)
        Method: Def-unit)
```

the PM below is generated:

```
(Derive Derived-Formula: u_1 * 1_U = u_1
        Method: Def-Semigroup*Unit)
```

The 15 preverbal messages generated by *PROVERB* for our example is listed (in machine format) below.

```
1. (Derive, Method: "hyp" ,
           Derived-Formula: ((GROUP F *) (SUBGROUP U F *)
                             (UNIT F 1 *) (UNIT U 1U)))
2. (Derive, Method: "Def-Semigroup*Unit" ,
           Derived-Formula: ((ELE 1U U)))
3. (Derive, Derived-Formula: ((EXISTS [X].(ELE X U))))
4. (Begin-Assume-Choice, Parameters: (X U) ,
                         Derived-Formula: ((ELE U1 U)))
5. (Derive, Method: "Def-Semigroup*Unit" ,
           Derived-Formula: ((== (APPLY * U1 1U) U1)))
6. (Derive, Reasons: ((SUBGROUP U F *))
           Derived-Formula: ((SUBSET U F)))
7. (Derive, Derived-Formula: ((ELE U1 F)))
8. (Similar, Reasons: ((ELE 1U U)),
           Derived-Formula: ((ELE 1U F)))
9. (Derive, Reasons: ((GROUP F *)),
           Derived-Formula: ((SEMIGROUP F *)))
10. (Derive, Reasons: ((== (APPLY * U1 1U) U)) ,
            Derived-Formula: ((SOLUTION U1 U1 1U F *)))
11. (Derive, Reasons: ((UNIT F 1 *)),
            Derived-Formula: ((== (APPLY * U1 1) U1)))
12. (Derive, Reasons: ((UNIT F 1 *)),
            Derived-Formula: ((ELE 1 F)))
13. (Derive, Reasons: ((ELE U1 F)),
            Derived-Formula: ((SOLUTION U1 U1 1 F *)))
14  (Derive, Method: "Th-solution",
            Derived-Formula: ((== 1U 1)))
15. (End-Assume-Choice, Parameters: (U),
                        Derived-Formula: ((== 1U 1)))
```

A sentence in English is generated from each PMs above by the surface generator TAG-GEN. The text as generated follows.

The Natural Language Proof

(1) Let F be a group and U be a subgroup of F and 1 be a unit element of F and 1_U be a unit element of U. (2) According to the definition of unit element $1_U \in U$. (3) Therefore there is an X, $X \in U$. (4) Now suppose that u_1 is such an X. (5) According to the definition of unit element $u_1 * 1_U = u_1$. (6) Since U is a subgroup of F $U \subset F$. (7) Therefore $u_1 \in F$. (8) Similarly $1_U \in F$ since $1_U \in U$. (9) Since F is a group F is a semigroup. (10) Since $u_1 * 1_U = u_1$ 1_U is a solution of the equation $u_1 * X = u_1$. (11) Since 1 is a unit element of F $u_1 * 1 = u_1$. (12) Since 1 is a unit element of F $1 \in F$. (13) Since $u_1 \in F$ 1 is a solution of the equation $u_1 * X = u_1$. (14) Since F is a group $1_U = 1$ by the uniqueness of solution. (15) This conclusion is independent of the choice of the element u_1.

Part III

Implementation

Chapter 13

The System PROVERB

13.1 Introduction and System Architecture

Based on the computational theories discussed in Part I and Part II, we have implemented a prototype of a proof presentation system called *PROVERB*. It is implemented in Allegro Common Lisp with CLOS, on top of KEIM a tool kit for constructing deduction systems [Nes92]. Although usable as a stand-alone system taking ND proofs as inputs, we have first experimented with it within Ω–MKRP, an interactive environment for developing proofs. A detailed overview of the proof and presentation cycle in Ω–MKRP is illustrated in Figure 13.1. The components within the dotted box are the main contributions of this thesis. Besides constructing a proof with predefined methods, a user of Ω–MKRP has also the possibility of calling an automated theorem prover to fill a remaining gap in the proof he is working on [HKK+92b]. Currently, the resolution based theorem prover MKRP is the only automated theorem prover fully integrated. The refutation graph delivered by MKRP is translated into ND proofs at the logic level, according to a mechanism reported in [Lin89, Lin90]. During this translation, emphasis is laid on the production of more natural proofs by avoiding indirect proofs and by globally structuring the proof. In this sense, it may be viewed as the first step of the reconstruction. Starting from a logic level proof the reconstruction is carried out one step further by abstracting it to the assertion level defined in Part I. Based on the computational model presented in Part II, a linearized list of preverbal messages is generated by a text planner. Finally, the preverbal messages are translated in a *template-driven* manner into syntactic and semantic descriptions accepted by Tag-Gen, a front generator based on an extended TAG formalism [Kil94, KF93]. In the next Section, we first discuss an algorithm which reconstructs an assertion level proof based on a logic level ND proof. Since the presentation of a logic level proof exactly follows the computational model discussed in Part II, we only illustrate this process with the help of an example in Section 13.3. In Section 13.4, the interface to Tag-Gen is briefly described.

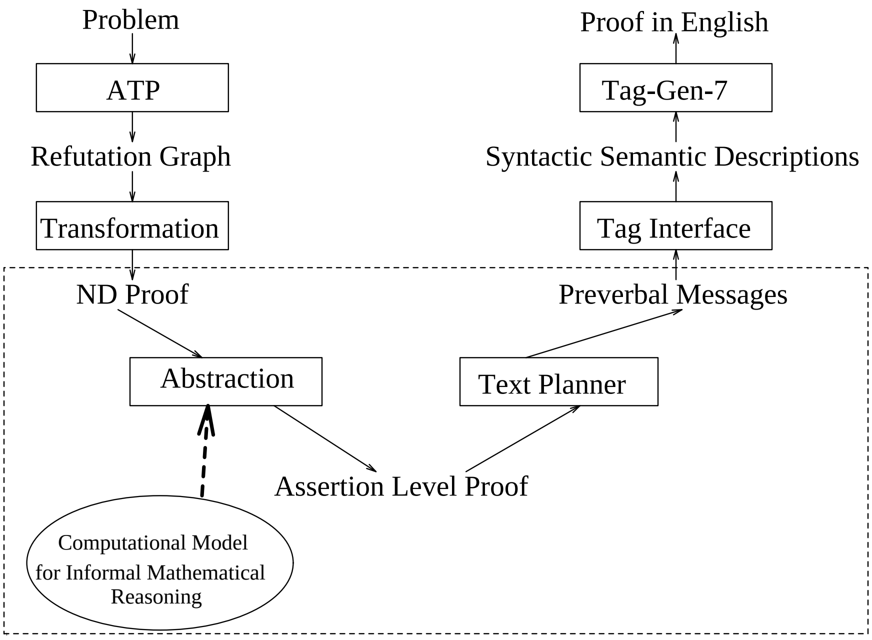


Figure 13.1: The Proof Presentation Cycle

13.2 The Abstraction Module

As sketched out in Chapter 1, a second reconstruction abstracting ND proofs to the assertion level is necessary, to provide the presentation module with more natural inputs. This section is devoted to such an abstraction procedure, with the help of the tree structure representing assertion level inference rules introduced in Chapter 7. Embedded in Ω–MKRP, an interactive proof development environment [HKK+92b], the input ND proofs are represented in a linearized version, introduced in [And80]. In this formalism, every proof is a sequence of proof lines, each of them is of the form:

$$Label \quad \triangle \quad \vdash \quad Derived\text{-}Formula \qquad Method(reason\text{-}pointers)$$

where *Method* is restricted to a rule of inference in $\mathcal{NK}$, which justifies the derivation of the derived formula using formulas in lines pointed to by reason-pointers as the preconditions. $\triangle$ is a finite set of formulas, being hypothesis on which the derived formula depends. We want to point out that although the linear format may not be suitable for some purposes, a linearization is required by the translation into natural language.

As argued above, in order to produce natural language proofs comparable with proofs found in typical mathematical textbooks, we should first try to replace as much complex proof units as possible by atomic assertion level steps. One straightforward procedure would be going through the entire input proof, and test for every proof line, if it can also be justified by the application of an assertion. As candidates for such assertions all formulas valid at this point of proof must be considered. To test the applicability of a particular assertion, we have to test for every compound rule associated with this assertion (compare Chapter 7), and to search the entire previous proof context for potential premises. Apparently, the above sketched procedure effectively reproves the problem based on the input proof. Although this procedure may find better proofs, it is very search intensive and can easily get lost in a combinatorially explosive search. Another extreme is a strict abstraction of the input ND proof by simply replacing all subproofs satisfying the decomposition and composition constraint (see section 6.1.1) by a atomic assertion level step. This approach, however, has a severe drawback as well. Since automated theorem provers usually work in a manner fundamentally different to that of human beings, the input ND proofs are often quite twisted so that not many units satisfying this constraint can be found. To reconcile the efficiency and the quality requirement, we employ an algorithm that mainly abstracts an existing proof as it is proved, but utilizes the assertion level inference rules instead of the decomposition and composition constraint. In fact, the global structure is taken over from the first approach: we go through the entire input proof, and test for every proof line, if it can also be justified by the application of an assertion. However, only definitions and theorems contributing to its proof, namely those in $\triangle$, are taken as candidates of applicable assertions. And only proof lines transitively reachable via the reason-pointers are considered as potential premises.

Algorithm:

1. go through the entire proof recursively starting from the conclusion, which is the root of the corresponding proof tress,
   (a) choose as the set of assertions AS the set of hypothesis $\triangle$ of the current line
   (b) among the lines transitively reachable via the reason-pointers starting from the current line, test if there exist lines $p_1, \ldots, p_n$, from which the derived formula F of the current line can be derived by applying an assertion $\mathcal{A}$ in AS. This is done by finding a subtree in $\mathit{Tree}(\mathcal{A}, \mathcal{NK})$, so that the derivation is justified by a rule represented by this subtree. In this case, replace the rule of inference $\mathcal{R}$ in this line by a label standing for $\mathcal{A}$ as the new justification, and update reason-pointers so that they point to $p_1, \ldots, p_n$. Recursively do 1. for $p_1, \ldots, p_n$.
2. delete all lines, which are no more involved in the reason hierarchy starting from the conclusion of the entire proof, along the reason-pointers.

A refinement is also made to tackle the situation where more than one applicable assertion level rule is found. In the current implementation, the one that deletes most lines is chosen. Although locally optimal choices do not always lead to a globally optimal choice, it turns out to be tolerable since such cases rarely occur. The search for a subtree in $\mathit{Tree}(\mathcal{A}, \mathcal{NK})$ is significantly accelerated by the *subformula relation* among nodes in the tree schemata (see 6.1.1).

Despite its simplicity, the current algorithm substantially shortens input ND proofs of a broad class. Most significant reduction is observed with input proofs which are essentially direct proofs, but containing machine-generated detours and redundancies. At the end of this section, we show an example where a machine-generated ND proof with 134 lines is abstracted to a proof of 15 lines. The algorithm also works well on neatly structured ND proofs. In these cases, the reduction factor depends on the average depth of the terms in the definitions and theorems involved in the proof. Since mathematicians usually avoid using both too trivial and too complicated definitions and theorems, a stable reduction factor (about two thirds in length) is normally achieved. The algorithm performs very poorly on machine-generated proofs which are mainly indirect, i.e., in most of the lines only contradiction is derived. Despite of a reduction factor of about one third in length, the remaining proof lines are still largely at the level of calculus rules and the proof is therefore still too tedious.

Let us look at the example below, abstracted from an input proof of 134 lines, generated in the proof development environment Ω–MKRP. Eleven of the remaining fifteen steps (from line 8 to line 21) are at the assertion level. The rest is justified by ND rules of more structural import: they introduce new temporary hypothesis or discharge them, or split a problem into cases. These steps will usually be presented explicitly later. Many trivial steps instantiating quantifiers or manipulating logical connectives are largely abstracted to assertion level steps. For instance, a proof segment with four extra lines, being the linearized version of the proof tree in Figure 2.1 as the matter of fact, is needed even in a neatly written input ND proof to achieve step 7 from step 2 and 5. Most importantly, the replacement is not restricted to natural expansions, but includes other logically equivalent compound segments. If we replace the assertion level steps in the proof below by their natural expansions correspondingly, the result is a logic level proof of 43 lines, in contrast to the input proof of 134 lines.

Satz 1.9 in [Deu71]:

Let F be a group and U a subgroup of F, if 1 and 1_U are unit elements of F and U respectively, then $1 = 1_U$.

The definitions of *semigroup*, *group*, and *unit* are obvious. $solution(a, b, c, F, *)$ should be read as "c is a solution of the equation $a * x = b$ in F."

Abstracted Proof about Unit Element of Subgroups

NNo	S;D		Formula	Reason
			Definitions	
1.	;1	$\vdash$	$\forall_{U,F}\ U \subset F \Leftrightarrow \forall_x\ x \in U \Rightarrow x \in F$	(Def-Subset)
2.	;2	$\vdash$	$\forall_{1,F,*}\ unit(F,1,*) \Leftrightarrow semigroup(F,*) \wedge 1 \in F \wedge (\forall_f\ f \in F \Rightarrow 1 * f = f * 1 = f)$	(Def-Semigroup*Unit)
3.	;3	$\vdash$	$\forall_{F,*}\ group(F,*) \Leftrightarrow semigroup(F,*) \wedge (\exists_1\ unit(F,1,*) \wedge (\forall_f\ f \in F \Rightarrow \exists_{f^{-1}}\ f^{-1} \in F \wedge f^{-1} * f = 1))$	(Def-Group)
4.	;4	$\vdash$	$\forall_{U,F,*}\ subgroup(U,F,*) \Leftrightarrow semigroup(F,*) \wedge U \subset F \wedge group(U,*)$	(Def-Subgroup)

5.	;5	$\vdash$	$\forall_{f,g,x,F,*}\ solution(f,g,x,F,*) \Leftrightarrow semigroup(F,*) \wedge f,g,x \in F \wedge f * x = g$	(Def-Solution)
6.	;6	$\vdash$	$\forall_{f,g,x,y,F,*}\ group(F,*) \wedge solution(f,g,x,F,*) \wedge solution(f,g,y,F,*) \Rightarrow x = y$	(Th-solution)
			The Proof	
7.	;7	$\vdash$	$group(F,*) \wedge subgroup(U,F,*) \wedge unit(F,1,*) \wedge unit(U,1_U,*)$	(Hyp)
8.	4;7	$\vdash$	$U \subset F$	(Def-Subgroup 7)
9.	2;7	$\vdash$	$1_U \in U$	(Def-Semigroup*Unit 7)
10.	2;7	$\vdash$	$\exists_x\ x \in U$	($\exists$ 9)
11.	;11	$\vdash$	$u_1 \in U$	(Hyp)
12.	2;7,11	$\vdash$	$u_1 * 1_U = u_1$	(Def-Semigroup*Unit 7 11)
13.	1,4;7,11	$\vdash$	$u_1 \in F$	(Def-Subset 8 11)
14.	1,2,4;7	$\vdash$	$1_U \in F$	(Def-Subset 8 9)
15.	3;7	$\vdash$	$semigroup(F,*)$	(Def-Group 7)
16.	1,2,3,4,5;7,11	$\vdash$	$solution(u_1,u_1,1_U,F,*)$	(Def-Solution 15 13 14 12)
17.	1,2,4;7,11	$\vdash$	$u_1 * 1 = u_1$	(Def-Semigroup*Unit 7 13)
18.	2;7	$\vdash$	$1 \in F$	(Def-Semigroup*Unit 7)
19.	1,2,3,4;7,11	$\vdash$	$solution(u_1,u_1,1,F,*)$	(Def-Solution 15 13 18 17)
20.	6,1,2,3,4,5;7,11	$\vdash$	$1 = 1_U$	(Th-Solution 7 16 19)
21.	1,2,3,4,5,6;7	$\vdash$	$1 = 1_U$	(Choice 10 20)
22.	1,2,3,4,5,6;	$\vdash$	$group(F,*) \wedge subgroup(U,F,*) \wedge unit(F,1,*) \wedge unit(U,1_U,*) \Rightarrow 1 = 1_U$	(Ded 7 21)

13.3 The Text Planner

The text planner of *PROVERB* expects as input the following:

- An ND style proof at the assertion level, such as the one listed in the last section.
- An ordered list of nodes indicating that the subproofs rooted at these nodes are subgoals when the proof was constructed.
- A list of features, with (compulsory specific general implicit explicit abstract detailed) as default. While (compulsory specific general) is a fixed order, the ordering of other features influences the global style of the text produced (compare 11.2).

Because several important presentation operators plan in terms of the subgoal structure formed during the proof construction time, *PROVERB* also accepts a list of nodes which are roots of proof units. Intuitively, proof units are subproofs whose conclusions are once posed explicitly as a subgoal during proof time. Thirdly, an ordered list of features is used to influence the overall style of the text produced. In the default case, a more implicit and abstract style is chosen. Usually, the sublist (*compulsory specific general*) should be included in each style. See Chapter 11 for further details of stylistic features.

Since it is simply an implementation of the computational model introduced in Part II, only two extra heuristic procedures are described here.

13.3.1 Heuristics Simulating Proof Time Subgoal Structure

Up to now, only part of the proof time subproof structure can be reconstructed in the transformation from refutation proofs to ND proofs [Lin90]. To relieve the user from always specifying the set of proof units manually as a parameter to the function `nl~presentation`, heuristics are devised to simulate it by structuring underlying mathematical theories into a hierarchy, where each theories is supported by its immediate subordinate theories. In general, we assume that definitions and theorems at a higher level in the hierarchy are more likely to play an important role in the proof process. As a first approximation, therefore, we assume that the subproofs rooted at nodes that are obtained from applying an assertion in the current theory. In the current system, the user is given the opportunity to define a theory by giving a list of assertions.

As the second type of plausible candidates, we also assume units involved in the application of *structural* natural deduction rules, since they determine the logical structure of the proofs. Concretely the current system employs the heuristics below:

Delineation of Proof Unit:

1. Every subtree in a proof tree rooted at a node derived by an inference rule which applies a definition or theorem of the current level is a proof unit.

2. In subproofs of the form $\dfrac{?Label_2 : \triangle, F \vdash G}{?Label_1 : \triangle, F \Rightarrow G} DEDuction$, mark both of the subtrees rooted at $?Label_1$ and $?Label_2$ as proof units.

3. In subproofs of the form $\dfrac{?Label_2 : \triangle, G \vdash \perp}{?Label_1 : \triangle \vdash \neg G} IP_1$, mark both of the subtrees rooted at $?Label_1$ and $?Label_2$ as proof units.

4. In subproofs of the form $\dfrac{Label_2 : \triangle \vdash F \vee G,\ \ ?Label_3 : \triangle, F \vdash H,\ \ ?Label_4 : \triangle, G \vdash H}{?Label_1 \triangle \vdash H} CASE$, mark the subtrees rooted at $?Label_1$, $?Label_2$, $?Label_3$, and $?Label_4$ as proof units.

5. In subproofs of the form $\dfrac{Label_2 : \triangle \vdash \exists_x Fx,\ \ Label_3 : \triangle, F_a \vdash H}{Label_1 : \triangle \vdash H} CHOICE$, mark the subtrees rooted at $Label_1$, $Label_2$, and $Label_3$ as proof units.

Note that proof unit structures related to the structural natural deduction rules are rarely used in the presentation process, since the structures duplicate with structures suggested by the top-down presentation operators relating to the corresponding proof structure, compare Section 11.3.

13.3.2 Combining Proof Communication Acts

Successive PCAs are sometimes combined in more powerful PCAs to produce more compact text. In *PROVERB* as it currently stands, two PCAs will only be combined if they are of the same type and share both the Reason slot and Method slot. The mechanism combining PCAs needs much more investigation and refinement.

13.4 The Integration of TAG-GEN as the Linguistic Component

To test the power of our computational model for proof presentation, in particular the appropriateness of the assertion level proof as an intermediate representation, TAG-GEN [Kil94, KF93] has been integrated as a linguistic component. TAG-GEN is a TAG-based syntactic generator that works in a parallel incremental way. Although developed as the surface generator for the multimodal presentation system WIP [WAB+92, WAF+93], it is aimed to suit a broad range of applications.

In particular, it fulfills the following requirements [Kil94]:

- incremental generation by utilizing parallelism,
- verbalization single utterances of varying size (sentences, phrases, words),
- accepting stylistic parameters to generate utterances of adequate styles,
- domain-independence with the help of declarative knowledge bases and flexible interfaces,
- multi-lingual output (currently English and German),

Our experience shows, that even with a simple translation mechanism, texts generated are of acceptable quality. Currently, preverbal messages are translated into a syntactic-semantic description language required by TAG-GEN in a template-driven way. The dictionary contains syntactic-semantic schemata both for PCAs and for predicate and function symbols of the first order predicate logic. For instance, below is the second preverbal message listed in Section 12.5.

```
(Derive, Method: "Def-semigroup*unit" ,
         Derived-Formula: ((ELE EU U)))
```

The corresponding tag-input generated is listed below.

```
(ADD-UTT-PAR :IDENTIFIER '|uttpar3064|)
(ADD-VP :HEAD "{\dcd $\in$}" :IDENTIFIER '|vp3067|
        :REGENT '|uttpar3064|
        :MOOD 'INDICATIVE :FOCUS '|mod-vp3066|)
(ADD-NP :HEAD "EU" :CAT 'NAME :GENDER 'NTR :SPECIFIER 'NONE
        :IDENTIFIER '|np3068| :REGENT '|vp3067|)
(ADD-NP :HEAD "U" :CAT 'NAME :GENDER 'NTR :SPECIFIER 'NONE
        :FUNC 'PATIENT :REGENT '|vp3067|)
(ADD-PP :HEAD "according to" :REGENT '|vp3067|
        :RELATION-IDENT '|mod-vp3066|
        :FUNC 'OPTIONAL :IDENTIFIER '|pp3069|)
(ADD-NP :HEAD "definition" :REGENT '|pp3069|
        :RELATION-IDENT '|mod-pp-13073|
         :FUNC 'PREPOBJECT :IDENTIFIER '|np3072|)
(ADD-PP :HEAD "of" :REGENT '|np3072|
        :RELATION-IDENT '|mod-pp-23075| :IDENTIFIER '|pp3071|)
(ADD-NP :HEAD "unit element" :CAT 'NAME :GENDER 'NTR
        :SPECIFIER 'NONE :REGENT '|pp3071|
        :RELATION-IDENT '|mod-pp-23074| :FUNC 'PREPOBJECT)
```

The sentence produced by TAG-GEN is:

> "According to the definition of unit element $E_U \in U$."

Note that the somewhat tedious syntax is determined by the incremental feature of Tag-Gen. The entire natural language Proof Generated by *PROVERB* for the subproof example discussed in Section 11.5 and Section 12.5 is given in Figure 9.7.

Chapter 14

Conclusion and Future Improvements

14.1 Conclusion

This thesis is centered around a reconstructive architecture for presenting machine-discovered proofs. This architecture finds its basis in two closely related computational theories: one about *informal mathematical reasoning*, and one about *human proof presentation*. The power of our architecture is derived to a great extent from the intermediate representation, namely, natural deduction (ND) style proofs at the *assertion level*. In contrast to original ND proofs, where the focus of attention is on syntactic manipulations, proofs reconstructed at the assertion level contain mostly inferences in terms of semantically meaningful operators, which apply a definition or a theorem valid in the context.

The computational model for informal mathematical reasoning is cast as an interleaving process of planning and verification, with the emphasis on more semantically meaningful plan operators that go beyond the level of primitive inference rules. By analyzing derivations involved in proofs in mathematical textbooks, we arrive at a formal definition of the intuitive notion of the application of a definition or a theorem, in terms of the structural characteristics exhibited in the corresponding natural deduction proofs. Definitions and theorems are collectively called *assertions*. Furthermore, the acquisition of compound inference rules at the assertion level is also described, leading to the notion of a *natural calculus*. With the help of a tree structure which represents compound inference rules in a compact way, machine-found ND proofs can be abstracted to the assertion level. The resulting proof is substantially shorter, and the remaining steps resemble the steps found in a typical mathematical textbook.

The presentation process itself is addressed in a computational model for human proof presentation, with a proof at the assertion level as the starting point. The most important feature of this model is that hierarchical planning and unplanned spontaneous presentation are integrated in the same framework in a complementary way. The top-down hierarchical planning views language generation as planned behavior. Based on explicit communicative knowledge encoded as schemata, it splits a presentation task into subtasks. Although our overall planning mechanism has much in common with the hierarchical planning pursued by RST-based text planners, the planning operators contain mostly complex presentation schemata associated with various proof structures. The bottom-up process aims at simulating the unplanned part of the proof presentation, where a still unpresented proof node is chosen as the next to be presented using only the local derivation relations. Since often more than one node is available via the local derivation relation, the local focus mechanism is employed to single out one of the candidates having the strongest semantic link with the focal centers. The reference choices affect significantly the quality of the text generated. By carefully analyzing proofs in mathematical textbooks, various forms for referring to inference methods and intermediate conclusions are identified. A discourse theory is designed to handle the *reference choices*. It benefits from a very natural segmentation of the discourse into an attentional hierarchy as the result of the distinction between the planned and the unplanned part of presentation.

Our theory for human proof presentation and a tractable algorithm for reconstructing assertion level proofs from logic level ND proofs is implemented in a prototypical system called *PROVERB*. It takes as input a natural deduction style proof and translates the proof in several steps into natural language. Currently, *PROVERB* is tested as an explanation component in a proof development environment called Ω–MKRP. The experience with this system demonstrates,, that acceptable natural language proofs can be produced for a wide range of ND proofs.

14.2 Future Improvements

Because the theories presented in this dissertation are intended to cover the entire spectrum of the presentation process, as well as a part of the proof construction process per se, refinements are necessary almost everywhere. Firstly, the reconstruction of proofs at the assertion level is only the first step toward presenting proofs beyond the level familiar from elementary logic. There is no doubt that proofs are often presented by mathematician at an even higher level of abstraction, since an expansion factor of 10 to 20 is reported when using systems like AUTOMATH [dB80] (i.e. the mechanical proofs are about 10 to 20 times longer than the corresponding human proof, whereas our "compression factor factor" is about 3 to 5 for neatly written ND proofs). Even more radical expansion factors (about 5,000 to 10,000) are conjectured by experts for harder mathematical problems, where only the top level proof schema is recorded and used for communication among mathematicians. To achieve a similar factor of reduction in the proof presentation, a plan recognition mechanism must be incorporated into the reconstruction architecture. The plan recognition in turn, must be based on a much deeper understanding of the cognitive process of informal mathematical reasoning.

In order to produce a coherent and flexible proof text, much refinement is necessary in the presentation model as well. First, since many presentation operators generate a complex scheme of PCAs, the resulting text may contain redundancies. Although a simple mechanism is integrated to combine successive PCAs, a more thorough solution might be to permit planning directly in terms of a single rhetorical relation or a single PCA, similar to the situation in RST-based planners. This might also help in the generation of cue words signalling the shift of global focus, which is currently lacking. Second, formulas are currently translated recursively in a template driven manner. If we examine proofs found in mathematical text books, the translations are normally more flexible. This problem is by no means simple, since the task of translation of a simple conjunctive formula is analogous to the description of a complex object [Par88]. According to our experience, the naturalness of the final verbalization relies heavily on the naturalness of the ontology chosen [GN87, LG90] as well as the standard translation associated with typical ontological structures. Third, closely related to the reconstruction at the plan level mentioned above, communicative norms concerning proof plan structures must be identified and encoded. Fourth, the current architecture normally generates only proofs in pure text that may contain formulas. A more advanced system must be able to generate formated text [AHvM91], as well as to include material in other modes as well, for instance, equations or even graphs separated from the text. To achieve this, assorted problems addressed in recent multimedia planners [WAB+92, FM90b, FM90a] must be investigated for mathematical application. In particular, it should be studied how material in different mode can be allocated [AHvM93]. Finally, proof presentation techniques are useful for tutoring systems. To provide a learning environment more flexible than mathematical textbooks, user modeling should be incorporated to tailor presentation for individual users.

Part IV

Appendixes

Appendix A

A Complete Example

Below is the complete presentation trace of the subset example used throughout Part II. The assertion level proof is repeated below for convenience.

Satz 1.9 in [Deu71]:

Let F be a group and U a subgroup of F, if 1 and 1_U are unit elements of F and U respectively, then $1 = 1_U$.

The definitions of *semigroup*, *group*, and *unit* are obvious. $solution(a, b, c, F, *)$ should be read as "c is a solution of the equation $a * x = b$ in F."

Abstracted Proof about Unit Element of Subgroups

NNo	S;D		Formula	Reason
			Definitions	
1.	;1	$\vdash$	$\forall_{U,F}\ U \subset F \Leftrightarrow \forall_x\ x \in U \Rightarrow x \in F$	(Def-Subset)
2.	;2	$\vdash$	$\forall_{1,F,*}\ unit(F, 1, *) \Leftrightarrow semigroup(F, *) \wedge 1 \in F \wedge (\forall_f\ f \in F \Rightarrow 1 * f = f * 1 = f)$	(Def-Semigroup*Unit)
3.	;3	$\vdash$	$\forall_{F,*}\ group(F, *) \Leftrightarrow semigroup(F, *) \wedge (\exists_1\ unit(F, 1, *) \wedge (\forall_f\ f \in F \Rightarrow \exists_{f^{-1}}\ f^{-1} \in F \wedge f^{-1} * f = 1))$	(Def-Group)
4.	;4	$\vdash$	$\forall_{U,F,*}\ subgroup(U, F, *) \Leftrightarrow semigroup(F, *) \wedge U \subset F \wedge group(U, *)$	(Def-Subgroup)
5.	;5	$\vdash$	$\forall_{f,g,x,F,*}\ solution(f, g, x, F, *) \Leftrightarrow semigroup(F, *) \wedge f, g, x \in F \wedge f * x = g$	(Def-Solution)
6.	;6	$\vdash$	$\forall_{f,g,x,y,F,*}\ group(F, *) \wedge solution(f, g, x, F, *) \wedge solution(f, g, y, F, *) \Rightarrow x = y$	(Th-solution)
			The Proof	
7.	;7	$\vdash$	$group(F, *) \wedge subgroup(U, F, *) \wedge unit(F, 1, *) \wedge unit(U, 1_U, *)$	(Hyp)
8.	4;7	$\vdash$	$U \subset F$	(Def-Subgroup 7)
9.	2;7	$\vdash$	$1_U \in U$	(Def-Semigroup*Unit 7)
10.	2;7	$\vdash$	$\exists_x\ x \in U$	($\exists$ 9)
11.	;11	$\vdash$	$u_1 \in U$	(Hyp)
12.	2;7,11	$\vdash$	$u_1 * 1_U = u_1$	(Def-Semigroup*Unit 7 11)

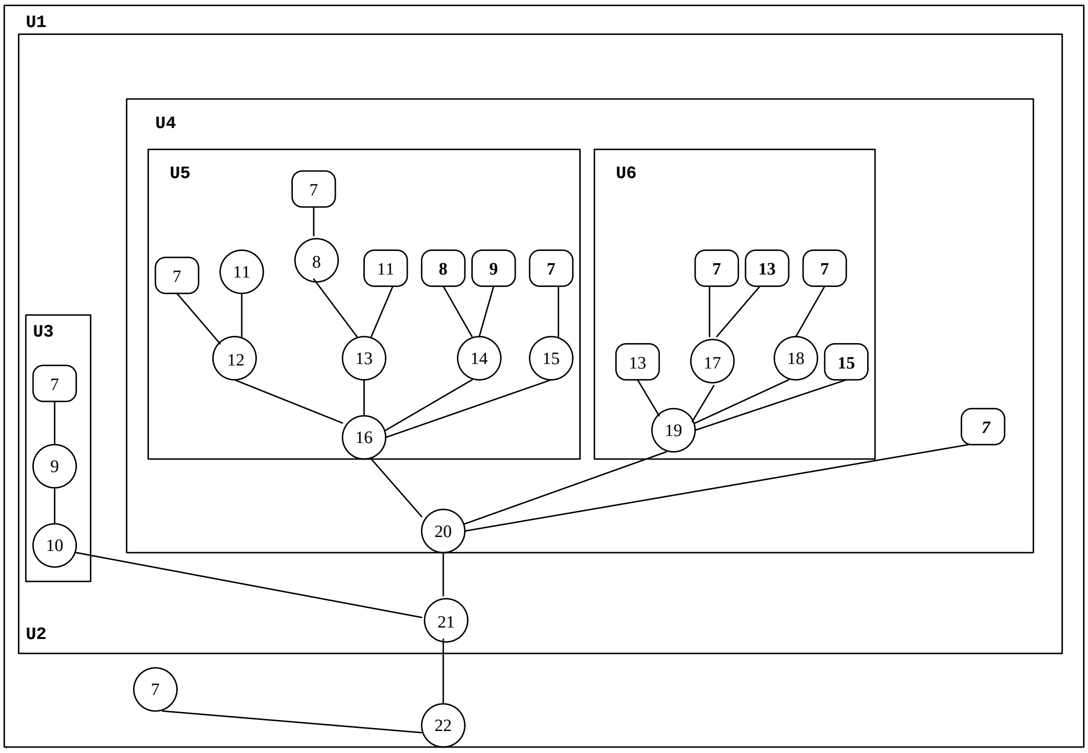


Figure A.1: Proof Tree for Satz 1.9

13.	1,4;7,11	$\vdash$	$u_1 \in F$	(Def-Subset 8 11)
14.	1,2,4;7	$\vdash$	$1_U \in F$	(Def-Subset 8 9)
15.	3;7	$\vdash$	$semigroup(F, *)$	(Def-Group 7)
16.	1,2,3,4,5;7,11	$\vdash$	$solution(u_1, u_1, 1_U, F, *)$	(Def-Solution 15 13 14 12)
17.	1,2,4;7,11	$\vdash$	$u_1 * 1 = u_1$	(Def-Semigroup*Unit 7 13)
18.	2;7	$\vdash$	$1 \in F$	(Def-Semigroup*Unit 7)
19.	1,2,3,4;7,11	$\vdash$	$solution(u_1, u_1, 1, F, *)$	(Def-Solution 15 13 18 17)
20.	6,1,2,3,4,5;7,11	$\vdash$	$1 = 1_U$	(Th-Solution 7 16 19)
21.	1,2,3,4,5,6;7	$\vdash$	$1 = 1_U$	(Choice 10 20)
22.	1,2,3,4,5,6;	$\vdash$	$group(F, *) \wedge subgroup(U, F, *) \wedge unit(F, 1, *) \wedge unit(U, 1_U, *) \Rightarrow 1 = 1_U$	(Ded 7 21)

Presentation Trace 1:

Initialization:

- Level of theories: definitions and proved theories are divided into several levels of theories, numbers below are line numbers in the proof above.
  - Set-Theory: 1
  - Group-Theory: 2, 3, 4

- Current-Theory: 5, 6

- Proof-time-subgoals: (10,16,19,20,21,22)
- Task-Stack:(22)

Notice, the numbering is the numbering of steps as used in Section 11.5

1. - OP: Preprocessor
   - PCA:

     (Derive Derived-Formula: $group(F,*) \ \wedge \ subgroup(U,F,*)$ $\wedge \ unit(F,1,*) \ \wedge \ unit(U,1_U,*)$
     Method: Hyp)

   - NL: "Let F be a group and U be a subgroup of F and 1 be a unit element of F and 1_U be a unit element of U."
   - Nodes-Conveyed: 7
   - Task-stack: (**21**)

2. - OP: Choice-Explicit-2
   - Task-stack: (**10**,20,21)

3. - Pre: (open-unit 10)(next-node 9)
   - OP: Derive-Bottom-Up
   - PCA:

     (Derive Derived-Formula: $1_U \in U$
     Reasons: $unit(1_U,U,*)$,
     Method: Def-Semigroup*Unit)

   - PM:

     (Derive Derived-Formula: $1_U \in U$
     Method: Def-Semigroup*Unit)

   - NL: "According to the definition of unit element, $1_U \in U$."
   - Nodes-Conveyed: 9
   - Task-stack: (**10**, 20, 21)

4. - Pre: (next-node 10)
   - OP: Derive-Bottom-Up

- PCA:

```
(Derive Derived-Formula: ∃_x x ∈U
        Reasons: 1_U ∈U
        Method: ∃I)
```

- PM: (Derive Derived-Formula: $\exists_x\ x \in U$)
- NL: "Therefore there is an X, $X \in U$."
- Nodes-Conveyed: 10
- post (close-unit 10)
- Task-stack: (20, 21)

5.
- Pre:
  - PCA: (Begin-Assume-Choice-1, u_1, x)
  - NL: "Now suppose that u_1 is such an X."
  - (Nodes-Conveyed: 11)
- OP: Proof-Time-Structure-Implicit
- Task-stack: (16,19,20,21)

6.
- Pre: (open-unit 16)(next-node 12)
- OP: Derive-Bottom-Up
- PCA:

```
(Derive Derived-Formula: u_1 * 1_U = u_1
        Reasons: (unit(1_U, U, *),  u_1 ∈U)
        Method: Def-Unit)
```

- PM:

```
(Derive Derived-Formula: u_1 * 1_U = u_1
        Method: Def-Unit)
```

- NL: "According to the definition of unit element, $u_1 * 1_U = u_1$."
- Nodes-Conveyed: 12
- Task-stack: (16,19,20,21)

7.
- Pre: (next-node 8)
- OP: Derive-Bottom-Up
- PCA:

```
(Derive Derived-Formula: U ⊂F
        Reasons: subgroup(U, F, *)
        Method: Def-Subgroup)
```

- PM:

```
(Derive Derived-Formula: U ⊂F
        Reasons: subgroup(U, F, *))
```

- NL: "Since U is a subgroup of F, $U \subset F$."
- Nodes-Conveyed: 8
- Task-stack: (16,19,20,21)

8.
- Pre: (next-node 13)
- OP: Derive-Bottom-Up
- PCA:

```
(Derive Derived-Formula: u_1 ∈F
        Reasons: u_1 ∈U,  U ⊂F
        Method: Def-Subset)
```

- PM: (`Derive Derived-Formula`: $u_1 \in F$)
- NL: "Therefore $u_1 \in F$."
- Nodes-Conveyed: 13
- Task-stack: (16,19,20,21)

9.
- Pre: (next-node 14)
- OP: Similar-Bottem-Up
- PCA:

```
(Similar-Assume-Derive Derived-Formula: 1_U ∈F
                       Reasons: 1_U ∈U,  U ⊂F
                       Methods:Def-Subset)
```

- PM:

```
(Similar-Assume-Derive Derived-Formula: 1_U ∈F
                       Reason: 1_U ∈U)
```

- NL: "Similarly $1_U \in F$, since $1_U \in U$."

- Nodes-Conveyed: 14
- Task-stack: (16,19,20,21)

10.
- Pre: (next-node 15)
- OP: Derive-Bottom-Up
- PCA:

```
(Derive Derived-Formula: semigroup(F, *)
        Reasons: group(F, *)
        Methods:Def-Group)
```

- PM:

```
(Derive Derived-Formula: semigroup(F, *)
        Reasons: group(F, *))
```

- NL: "Since F is a group, F is a semigroup."
- Nodes-Conveyed: 15
- Task-stack: (16,19,20,21)

11.
- Pre: (next-node 16)
- OP: Derive-Bottom-Up
- PCA:

```
(Derive Derived-Formula: solution(u_1, u_1, 1_U, F, *)
        Reasons: Node12, Node13, Node14, Node15
        Methods:Def-Solution)
```

- PM:

```
(Derive Derived-Formula: solution(u_1, u_1, 1_U, F, *)
        Reasons: Node12)
```

- NL: "Since $u_1 * 1_U = u_1$, 1_U is a solution of the equation $u_1 * X = u_1$."
- Nodes-Conveyed: 16
- post (close-unit 16)
- Task-stack: (19,20,21)

12.
- Pre: (open-unit 19)(next-node 17)
- OP: Derive-Bottom-Up
- PCA:

```
(Derive Derived-Formula: u_1 * 1 = u_1
        Reasons: unit(F,1,*)
        Methods:Def-Unit)
```

- PM:

```
(Derive Derived-Formula: u_1 * 1 = u_1
        Reasons: unit(F,1,*)
```

- NL: "Since 1 is a unit element of F, $u_1 * 1 = u_1$."
- Nodes-Conveyed: 17
- Task-stack: (19,20,21)

13.
- Pre: (next-node 18)
- OP: Derive-Bottom-Up
- PCA:

```
(Derive Derived-Formula: 1 ∈F
        Reasons: unit(F,1,*),
        Methods:Def-Unit)
```

- PM:

```
(Derive Derived-Formula: 1 ∈F
        Reasons: unit(F,1,*))
```

- NL: "Since 1 is a unit element of F, $1 \in F$."
- Nodes-Conveyed: 18
- Task-stack: (19,20,21)

14.
- Pre: (next-node 19)
- OP: Derive-Bottom-Up
- PCA:

```
(Derive Derived-Formula: solution(u_1,u_1,1,F,*)
        Reasons: Node13, Node17, Node18, Node15
        Methods:Def-Solution)
```

- PM:

```
(Derive Derived-Formula: solution(u_1,u_1,1,F,*)
        Reasons: Node13)
```

- NL: "Since $u_1 \in F$, 1 is a solution of the equation $u_1 * X = u_1$."
- Nodes-Conveyed: 19
- post: (close-unit 19)
- Task-stack: (20,21)

15.
- Pre: (next-node 20)
- OP: Derive-Bottom-Up
- PCA:

```
(Derive Derived-Formula: 1 = 1_U
        Reasons: Node16, Node19, Node7
        Methods:Th-Solution)
```

- PM:

```
(Derive Derived-Formula: 1 = 1_U
        Reasons: Node7
        Methods:Th-Solution)
```

- NL: "Since F is a group, $1_U = 1$ by the uniqueness of solution."
- Nodes-Conveyed: 20
- post:
  - PCA: (End-Assume-Choice-1 u_1)
  - NL: "This conclusion is independent of the choice of the element u_1."
  - (close-unit 20)(set-conveyed 21,22)
- Task-stack: ()

List of PM's

```
1. (Derive, Method: "hyp" ,
           Derived-Formula: ((GROUP F *) (SUBGROUP U F *)
                             (UNIT F 1 *) (UNIT U 1U)))
2. (Derive, Method: "Def-Semigroup*Unit" ,
           Derived-Formula: ((ELE 1U U)))
3. (Derive, Derived-Formula: ((EXISTS [X].(ELE X U))))
4. (Begin-Assume-Choice, Parameters: (X U) ,
                         Derived-Formula: ((ELE U1 U)))
5. (Derive, Method: "Def-Semigroup*Unit" ,
           Derived-Formula: ((== (APPLY * U1 1U) U1)))
6. (Derive, Reasons: ((SUBGROUP U F *))
           Derived-Formula: ((SUBSET U F)))
7. (Derive, Derived-Formula: ((ELE U1 F)))
8. (Similar, Reasons: ((ELE 1U U)),
           Derived-Formula: ((ELE 1U F)))
9. (Derive, Reasons: ((GROUP F *)),
           Derived-Formula: ((SEMIGROUP F *)))
10. (Derive, Reasons: ((== (APPLY * U1 1U) U)) ,
            Derived-Formula: ((SOLUTION U1 U1 1U F *)))
11. (Derive, Reasons: ((UNIT F 1 *)),
            Derived-Formula: ((== (APPLY * U1 1) U1)))
12. (Derive, Reasons: ((UNIT F 1 *)),
            Derived-Formula: ((ELE 1 F)))
13. (Derive, Reasons: ((ELE U1 F)),
            Derived-Formula: ((SOLUTION U1 U1 1 F *)))
14  (Derive, Method: "Th-solution",
            Derived-Formula: ((== 1U 1)))
15. (End-Assume-Choice, Parameters: (U),
                        Derived-Formula: ((== 1U 1)))
```

The Natural Language Proof

(1) Let F be a group and U be a subgroup of F and 1 be a unit element of F and 1_U be a unit element of U. (2) According to the definition of unit element $1_U \in U$. (3) Therefore there is an X, $X \in U$. (4) Now suppose that u_1 is such an X. (5) According to the definition of unit element $u_1 * 1_U = u_1$. (6) Since U is a subgroup of F $U \subset F$. (7) Therefore $u_1 \in F$. (8) Similarly $1_U \in F$ since $1_U \in U$. (9) Since F is a group F is a semigroup. (10) Since $u_1 * 1_U = u_1$ 1_U is a solution of the equation $u_1 * X = u_1$. (11) Since 1 is a unit element of F $u_1 * 1 = u_1$. (12) Since 1 is a unit element of F $1 \in F$. (13) Since $u_1 \in F$ 1 is a solution of the equation $u_1 * X = u_1$. (14) Since F is a group $1_U = 1$ by the uniqueness of solution. (15) This conclusion is independent of the choice of the element u_1.

Appendix B

A Library of PCAs

This appendix is a complete listing of the PCAs employed in our system. To form a PM, slots *Reasons*, *Intermediate-Formulas* and *Method* of a PCA should be augmented by one of the three status: *explicit*, *implicit* or *omit*. To use Tag-Gen as the surface generator, a dictionary is employed mapping patterns of PMs into corresponding syntactic structure. Therefore, what can be found below is a listing of possible patterns of PMs, each associated with one or more verbalizations. Note that, only the slots with explicit status are included.

1. (Assume-1 $Formulas$)

   Suppose $Formulas$[1]

2. (Axiom-Instantiation-1 $Formulas$)

   Let $Formulas$, Now suppose that $Formulas$

3. (Begin-Assume-Choice-1 a, b)

   Let b be such an a

4. (Begin-Assume-Choice-2 $Formula$)

   Suppose that $Formula$,

5. (Begin-Cases)

   (a) (Begin-Cases)

      Let us consider the following two cases[2].

   (b) (Begin-Cases Goal: *Formula-1*
   Assumptions: (*Assumption-1 Assumption-2*))

      To prove $Formula_1$, let us consider the cases that *Assumption-1* and the case that *Assumption-2*

[1] In verbalizations, the formulas stand for their verbalizations

[2] The case rule in Ω–MKRP currently handles just two cases, compare Chapter 4. This PCA can be generalized to handle n cases in a straightforward way.

6. (Case-First)
   (a) (Case-First)
   Let us first consider the first the case.
   (b) (Case-First *Formula*)
   First, let us consider the first case by assuming *Formula*
7. (Case-Second)
   (a) (Case-Second)
   Next let us consider the second case.
   (b) (Case-Second *Formula*)
   Next, we consider the second case by assuming *Formula*
8. Derive:
   (a) (Derive Derived-Formula: *Formula-1*
   Intermediate-Formulas: *List-of-formulas*
   Method: *Assertion-as-formula*
   Therefore, since (because) *List-of-formulas* and *Assertion-as-formula*, *Formula-1*.
   (b) (Derive Derived-Formula: *Formula-1*)
   Clearly (thus, hence), *Formula-1*
   (c) (Derive Derived-Formula: *Formula-1*
   Intermediate-Formulas: *List-of-formulas*
   Method: *name-of-assertion*
   Since (because) *List-of-formulas*, according to *name-of-assertion*, *Formula-1*.
   (d) (Derive Derived-Formula: *Formula-1*
   Reasons: *List-of-formulas*
   Method: *name-of-assertion*)
   Since (because) *List-of-formulas*, according (by) to *name-of-assertion*, *Formula-1*.
   (e) (Derive Derived-Formula: *Formula-1*
   Reasons: *List-of-formulas-1*
   Intermediate-Formulas: *List-of-formulas-2*)
   Since *list-of-formulas-1*, this means *list-of-formulas-2* and therefore *formula-1*.
   (f) (Derive Derived-Formula: $\perp$)
   This is a contradiction. This leads to a contradiction.
9. (End-Assume-Choice-1 a)
   This conclusion is independent of the choice of the element a.

10. (Proof-By-Contradiction-1)

    We prove by contradiction.

11. (New-Lemma *Formula*)

    Let's first prove *Formula* as a lemma.

12. Reformulate

    (a) (Reformulate Derived-Formula: *Formula-1*
    Reasons: *List-of-Formulas*)

    Since *Formula-1*, in other words, *List-of-Formulas*;

    (b) (Reformulate Derived-Formula: *Formula-1*)

    In other words, *Formula-1*.

13. Similarly

    (a) (Similar-Assume-Derive Assumption: Formula-1
    Derived-Formula: *Formula-2*
    Intermediate-Formulas: *Formula-list*)

    Similarly, we derive *Formula-List* by assuming *Formula-1*, which leads to *Formula-2*.

    (b) (Similar-Derive Derived-Formula: *formula*
    Reasons: *Formula-List*)

    Similarly, since *Formula-List*, *formula*.

14. (Split-goal Goal: *Formula-1*
    Subgoals: *List-of-Formulas*)

    To prove *Formula-1*, Let's first show that *List-of-formulas*.

Appendix C

A Library of Presentation Operators

C.1 Top-Down Planning Operators

Assertion-Implicit

- Proof: $\dfrac{?Label_1 : subproof_1, \ldots, ?Label_n : subproof_n}{?Label_{n+1} : A \vdash Q}\, ?M$

- Applicability Condition: $((\exists_i$ local-focus $?Label_i) \vee$ (task $?Label_{n+1}))$
$\wedge(\exists_i$ (not-conveyed $?Label_i)) \wedge$ (assertion $?M$)

- Acts:

```
(order the ?label_i in an order intrinsic to   ?M)
(go through the above ordered list
    (if (not-conveyed ?Label_i)
        (present Label_i)))
```

- Features: (top-down compulsory implicit detailed)

One of the focus guided top-down operators, implicit style. The explicit version of this operator is omitted here.

Assume-Choice

- Proof: $$\dfrac{?Label_2 : \triangle \vdash \exists_x P_x, Hyp, \qquad \dfrac{\dfrac{?Label_4 : P_a \vdash P_a}{\vdots}}{?Label_3 : \triangle, P_a \vdash R}}{?Label_1 : \triangle \vdash R}\text{Choice}$$

- Applicability Condition: (task ?$Label_1$) $\wedge$ (conveyed ?$Label_2$) $\wedge$ (not-conveyed ?$Label_4$)
- Acts:

```
(if (local-focus ?Label2)
    (Begin-Assume-Choice-1, x, a)
    (Begin-Assume-Choice-2, ?Label4))
(mark-conveyed ?Label2)
(Present ?Label3)
(End-Assume-Choice-1 a)
(mark-conveyed Label1))
```

- Features: (top-down compulsory explicit detailed)

Explicit choice operator, when it is reached in a top-down way, but $Label_2$ is already conveyed.

Case-Explicit

- Proof: as given in Figure 11.1.
- Applicable Condition: ((task ?$Label_1$) $\vee$ (local-focus ?$Label_4$)) $\wedge$ (not-conveyed (?$Label_2$, ?$Label_3$))
- Acts:

```
(cond ((not-conveyed ?Label4)              ;;;first case
       (Subgoal Label1 (?Label4?Label2 ?Label3))
       (Present ?Label4))
      (T (Subgoal ?Label1 (?Label2, ?Label3)))) ;;;second case
(Case-First, P)
(present ?Label2)
(Case-Second, Q)
(present ?Label3)
(set-conveyed ?Label1))
```

- Features: (top-down compulsory explicit)

Below is the implicit due of the operator above.

Case-Implicit

- Proof: as given in Figure 11.1
- Applicability Condition: ((task ?$Label_1$) $\vee$ (local-focus ?$Label_4$)) $\wedge$ (not-conveyed (?$Label_2$?$Label_3$))
- Acts:

```
(if (not-conveyed ?Label4) ;;;Labels stand for
    (present ?Label4))     ;;;the corresponding nodes
(Case-First F)
(present ?Label2)
(Case-Second G)
(present ?Label3)
(set-conveyed ?Label1))
```

- features: (top-down compulsory implicit)

Case-Complementary

- Proof: $$\dfrac{?Label_2 : \triangle \vdash P \vee \neg P; \quad \dfrac{\dfrac{?Label_5 : P \vdash P}{\vdots}}{?Label_3 : \triangle, P \vdash Q}; \quad \dfrac{\dfrac{?Label_6 : \neg P \vdash \neg P}{\vdots}}{?Label_4 : \triangle, \neg P \vdash Q}}{?Label_1 : \triangle \vdash Q}\text{Case}$$

- Applicability Condition: (task $?Label_1$) $\vee$ (unconveyed $?Label_3$, $?Label_4$)

- Acts:

```
(mark-conveyed ?Label2)
(Case-First Label5)
(Present ?Label3)
(Case-Second Label6)
(Present ?Label4)
(mark-conveyed ?Label1)
```

- Features: (top-down compulsory explicit detailed)

The case rule is not explicitly conveyed, if the two cases are complementary

The communication norm relating to the structural Gentzen rule Choice is encoded in the following two presentation operators.

Choice-Explicit-1

- Proof: $$\cfrac{?Label_2 : \triangle \vdash \exists_x P_x, \qquad \cfrac{\cfrac{?Label_4 : P_a \vdash P_a}{\vdots}}{?Label_3 : \triangle, P_a \vdash R}}{?Label_1 : \triangle \vdash R}\text{Choice}$$

- Applicability Condition:(task $?Label_1$) $\vee$ (father $?Label_1$, local-focus) $\wedge$ not-conveyed($?Label_3$)

- Acts:

```
(if (not-conveyed ?Label2)
    (present ?Label2))
(Begin-Assume-Choice a, x)
(present ?Label3)
(End-Assume-Choice-1 a)
(mark-conveyed ?Label1)
```

- Features: (top-down compulsory explicit detailed)

Explicit choice operator, when the proof is reached in a bottom-up way.

Choice-Explicit-2

- Proof: $$\cfrac{?Label_2 : \triangle \vdash \exists_x P_x, \qquad \cfrac{\cfrac{?Label_4 : P_a \vdash P_a}{\vdots}}{?Label_3 : \triangle, P_a \vdash R}}{?Label_1 : \triangle \vdash R}\text{Choice}$$

- Applicability Condition:(not-conveyed ($?Label_2$ $?Label_3$)) $\wedge$ (task $?Label_1$)

- Acts:

```
(Present ?Label2)
(Begin-Assume-Choice a, x)
(mark-conveyed ?Label4)
(present ?Label3)
(End-Assume-Choice-1 a)
```

- Features: (top-down compulsory explicit detailed)

Explicit choice operator, when the proof is reached in a top-down way. The implicit counterpart of the operators above is omitted.

Deduction-Top-Down

- Proof: $$\dfrac{\dfrac{\dfrac{?Label_1 : \triangle, P \vdash P, Hyp}{\vdots}}{?Label_n : \triangle, P \vdash Q}}{?Label_{n+1} : \triangle \vdash P \Rightarrow Q}\text{DED}$$

- Applicability Condition: (task $?Label_{n+1}$) $\wedge \exists_i$ (not-conveyed $?Label_i$)

- Acts:

```
(Derive Derive-Formula P)
(present ?Label_n)
(set-conveyed ?Label_{n+1})
```

- Features: (top-down compulsory implicit detailed)

Top-down planning by first establish the assumption

Below, we going to examine communications norms for some other proof structures. The following presentation operator captures the way indirect proof is normally presented. The PCAs (Proof-By-Contradiction-1) and (Assume-1 $?Label_1$) will generate "We prove by contradiction." and "Let us assume $\neg P$.".

Indirect-Proof-Explicit

- Proof: $$\dfrac{\dfrac{\dfrac{?Label_3 : \neg P \vdash \neg P}{\vdots}}{?Label_2 : \triangle, \neg P \vdash \perp}}{?Label_1 : \triangle \vdash P}\text{IP}$$

- Applicability Condition: (not-conveyed $?Label_3$) $\wedge$ (task $?Label_1$)

- Acts:

```
(Proof-By-Contradiction-1)
(Assume-1, Formulas: ?Label_3)
(set-conveyed ?Label_3)
(present ?Label_2)
(present ?Label_1)
```

- Features: (top-down compulsory explicit detailed)

The implicit dual of the operator above is below.

Indirect-Proof-Implicit

- Proof: $$\cfrac{\cfrac{?Label_3 : \neg P \vdash \neg P}{\vdots}}{\cfrac{?Label_2 : \triangle, \neg P \vdash \bot}{?Label_1 : \triangle \vdash P}\text{IP}}$$

- Applicability Condition: (not-conveyed $?Label_3$) $\wedge$ (task $?Label_1$)

- Acts:

  (Proof-By-Contradiction-1)
  (Assume-1, Formulas: $?Label_3$)
  (set-conveyed $?Label_3$)
  (present $?Label_2$)
  (set-conveyed $?Label_1$)

- Features: (top-down compulsory implicit detailed)

Lemma

- Applicability Condition: (not-conveyed $?Label_1$) $\wedge$ (in-unit active-unit $?Label_1$) $\wedge$ (important-subproof $?Label_1$)

- Acts:

  (New-Lemma $?Label_1$)
  (present $?Label_1$)

- Feature: (top-down specific explicit detailed)

Establishes a new lemma as a subgoal, and sets out to present it.

Apart from the presentation norms, the presentation process is also strongly influenced by the proof searching process per se, in particular the subgoal structure. For example, given the proof in Fig. C.1, where node A, B2 are once subgoals for the proof planer. The presentation process, preferring an implicit style, might determine to present the task-1, task-2 and the task-3 subsequently.

More generally, suppose S is a proof with subproofs $S_1, \ldots, S_n$, whose roots were all once a subgoal for the proof planning process. The presentation process will travel the subgoal structure in an an appropriate order and makes them the presentation goals in turn. The ordering of the subgoals is determined in the operator **Proof-Time-Structure-Implicit** by applying the three ordering principles described in Chapter 12 in the following order:

- the *proof time* order principle,

- the *local focus* principle,

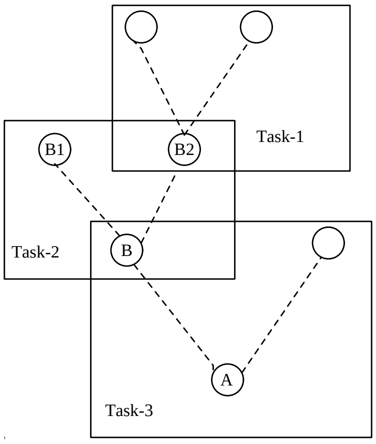


Figure C.1: A subgoal example

- the *minimal load* principle.

The following is the corresponding presentation operator based on proof time structure.

Proof-Time-Structure-Implicit

- Proof: a proof P with $P_1, \ldots, P_n$ as partially ordered subgoals in the subgoal hierarchy.
- Applicability Condition: (task P)
- Acts:

```
(cond ((the proof time order of subproofs of P  ;ordering
         is important and available)                   ;case 1
        (order the subgoals using proof time information))
       ((if the subgoals are still not totally ordered) ;case 2
        (order the subgoals using the focus principle))
       ((if the subgoals are still not totally ordered) ;case 3
        (order the subgoals using the minimal load principle)))
(for i from 1 to n (if (not presented Pi)   ;presenting
                        (present Pi)))
```

- Features: (top-down general implicit detailed)

Divide into subproofs according to proof time structure, focus principle, minimal load principle.

Split-Top-Down-Implicit

- Proof: a proof P with $P_1, \ldots, P_n$ as ordered subgoals in the subgoal hierarchy.
- Applicability Condition: (task P)

- Acts:

```
(for i:=1 to n
    (if (not-conveyed Pi)
        (present Pi)))
```

- Features: (top-down general implicit detailed)

Order-Minimal-Load

- Proof: a proof P with $P_1, \ldots, P_n$ as unordered subproofs.
- Applicability Condition: (task P)
- Acts: (order the subgoals reverse to the size of the subproofs)
- Features: (top-down general implicit)

Order subproofs according to minimal load principle.

Order-Focus

- Proof: a proof P with $P_1, \ldots, P_n$ as unordered subproofs.
- Applicability Condition: (task P)
- Acts: (order the subproofs by repeatedly applying function `focus-select-unit` starting from the local focus)
- Features: (top-down general)

```
(def focus-select-unit (U: set of units, P: an assertion)
     (until (P is exclusively in in one Ui ∈ U)
            (setq P (Focus-Select P)))
     (return Ui))
```

Order-Proof-Time

- Proof: a proof P with $P_1, \ldots, P_n$ as unordered subproofs.
- Applicability Condition: (task P)
- Acts: (order the subproofs according to the proof-time order)
- features: (top-down specific)

C.2 Bottom-Up Operators

Axiom-Instantiation

- Proof: $\dfrac{}{?Node}$Hyp,
- Applicability Condition: (eq next-node $?Node$) $\wedge$ (axiom-in-KB $?Node$)
- Act: (Axiom-Instantiation, Formulas: $?Node$)
- Features: (bottom-up compulsory explicit detailed)

Instantiate a axiom schema in first order logic.

Derive-Bottom-Up

- Proof: $\dfrac{?Node_1, \ldots, ?Node_n}{?Node_{n+1}}?M$
- Applicability Condition: (eq next-node $?Node_{n+1}$) $\wedge$ (conveyed ($?Node_1$,...,$?Node_n$))
- Acts:

```
(Derive Derived-Formula: ?Node_{n+1}
        Reasons: ( ?Node_1,···, ?Node_n)
        Method: ?M)))
```

- Features: (bottom-up general explicit detailed)

Deduction

- Proof: $\dfrac{?Label_1 : \triangle, P \vdash Q}{?Label_2 : \triangle \vdash P \Rightarrow Q}$DED
- Applicability Condition: (eq next-node $?Label_2$) $\wedge$ (conveyed $?Label_1$)
- Acts:

```
(mark-conveyed ?Label_2)
```

- Feature: (bottom-up compulsory implicit)

Deduction rule in bottom-up manner, no speech act is needed.

Reformulation-Implicit-1

- Proof: proof as in Figure 11.4

- Applicability Condition: (in-unit $?Q_n$ current-unit) $\wedge$ (eq next-node $?Q_1$)) $\wedge\forall_{i=1\ to\ m}$ (conveyed $?P_m$)$\wedge\forall_{i=1\ to\ n}$ (reformulation $?M_i$)

- ACTS:

```
(cond ((important ?Qn)       ;;;case 1
       (Derive Derived-Formula: ?Qn
               Reasons: ?P1, ···, ?Pm
               Method: ?M1))
      ((> n 4)               ;;;case 2
       (Derive Derived-Formula: ?Q1
                      Reasons: ?P1, ···, ?Pm
                      Method: ?M1)
       (Reformulate Derived-Formula: ?Qn))



      (T (Derive Derived-Formula: ?Q1 ;;;case 3
                  Reasons: ?P1, ···, ?Pm
                  Method: ?M1))
```

- features (bottom-up compulsory explicit abstract)

Similar-Assume-Bottom-Up

- Proof: $$\frac{\dfrac{?Label_1:\ P_1,\ldots,(?Label_i:\ A\vdash A\ Hyp),\ldots Label_n:\ P_n}{\vdots}}{?Label_{n+1}:\ \triangle\vdash Q}$$

- Applicability Condition: (in-unit current-unit $?Label_1$) $\wedge\forall_{j:1\ to\ (i-1),(i+1)\ to\ n}$ (conveyed $Label_i$) $\wedge$ (eq next-node $Label_i$)$\wedge$ a subproof considered similar to the one rooted by $?Label_1$ was just presented.

- Acts:

```
(setq inter important-intermediate-formulas)
(Assume-Derive-Similar Assumption: A,
                       Derived-Formula: Q,
                       Intermediate-Formula: inter))
```

- Features: (bottom-up specific explicit abstract)

In the current implementation, the detection of similarities is carried out by a heuristic function, no important intermediate results is chosen.

Similar-Proof-Bottom-Up

- Proof: $\dfrac{?Node_1, \ldots, ?Node_n}{\dfrac{\vdots}{?Node_{n+1}}}$
- Applicability Condition: (in-unit current-unit $?Node_{n+1}$) $\wedge$ (in-subtree next-node $Node_{n+1}$) $\wedge \forall_{i=1 \ to \ n}$ (conveyed Label$_i$) $\wedge$ (a subproof considered similar to the one rooted by $?Node_1$ was just presented.)
- Acts:

  (Similar Derived-Formula: $Node_{n+1}$,
  Reasons: $?\ Node_1, \ldots, ?\ Node_n$))
- Features (bottom-up specific explicit abstract)

Simplify-Bottom-Up

- Proof: $\dfrac{?Node_1, \ldots, ?Node_n}{\dfrac{\vdots}{?Node_{n+1}}}$
- Applicability Condition: (in-unit active-unit $?Node_{n+1}$) $\wedge$ (in-subproof next-node $Node_{n+1}$) $\wedge \forall_{i=1 \ to \ n}$ (conveyed $?Node_i$) $\wedge$ (simple-proof $?Node_{n+1}$)
- Acts:

  (Derive Derived-Formula: $?Node_{n+1}$,
  Reasons: $?\ Node_1, \ldots, ?\ Node_n$)
- Features (bottom-up general explicit abstract)

The simplicity of a proof is currently judged by a heuristic function which takes both number of nodes and the type of the nodes in the proof into account.

Index

Bibliography

[AHvM91] Yigal Arens, Eduard Hovy, and Susanne van Mulken. Automatic generation of formatted text. In *Proc. of AAAI-91*, pages 92–97, Anaheim, 1991. Morgan Kaufmann.

[AHvM93] Yigal Arens, Eduard Hovy, and Susanne van Mulken. Structure and rules in automated mulimedia presentation planning. In Ruzena Bajcsy, editor, *Proc. of IJCAI-93*, volume 2, pages 1253–1259, Chambéry, France, 1993. Morgan Kaufmann.

[And80] Peter B. Andrews. Transforming matings into natural deduction proofs. In *Proc. of the 5th International Conference on Automated Deduction*, pages 281–292. Springer, 1980.

[And86] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: to Truth Through Proof.* Academic Press, 1986.

[Bla81] L. H. Blaine. Programs for structured proofs. In P. Suppes, editor, *University-level Computer-assisted Instruction at Stanford: 1968-1980.* Institute for Mathematical Studies in the Social Sciences, Stanford University, Stanford, California, 1981.

[Bra78] Martin D.S. Braine. On the relation between the natural logic of reasoning and standard logic. *Psychological Review*, 85(1):1–21, Jan. 1978.

[Bra79] Ronald J. Brachman. On the epistemological status of semantic networks. In Nicholas V. Findler, editor, *Associative Networks.* Academic Press, 1979.

[Bun88] Alan Bundy. The use of explicit plans to guide inductive proofs. In *Proc. of 9th International Conference on Automated Deduction*, pages 111–120. Springer, 1988.

[Bun90] Alan Bundy. A science of reasoning: Extended abstract. In Mark E. Stickel, editor, *Proc. of 10th International Conference on Automated Deduction*, pages 633–640. Springer, 1990.

[CAB+86] R.L. Constable, S.F. Allen, H.M. Bromley, W.R. Cleaveland, J.F. Cremer, R.W. Harper, D.J. Howe, T.B. Knoblock, N.P. Mendler, P. Panangaden, J.T. Sasaki, and S.F. Smith. *Implementing Mathematics with the Nuprl Proof Development System.* Prentice Hall, New Jersey, 1986.

[Cha75] W. Chafe. Structures and human knowledge. In J. B. Carrol and R. O. Freedle, editors, *Language Comprehension and the Acquisition of Knowledge.* Books Demand UMI Publications, Charlotte, N.C., 1975.

[Cha76] W. Chafe. Giveness, constrastiveness, definiteness, subjects and topics. In C. Li, editor, *Subject and Topic.* Academic Press, 1976.

[Che76] Daniel Chester. The translation of formal proofs into English. *AI*, 7:178–216, 1976.

[Dal88] Robert Dale. The generation of subsequent referring expressions in structured discourses. In G. Sabah M. Zock, editor, *Advances in Natural Language Generation*, pages 58–75. 1988.

[Dal92] Robert Dale. *Generating Referring Expressions.* ACL-MIT PressSeries in Natural Language Processing. MIT Press, 1992.

[dB80] Nicolaas Govert de Bruijn. A survey of the project AUTOMATH. In J. P. Seldin and J. R. Hindley, editors, *To H.B. Curry - Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 579–606. Academic Press, 1980.

[Deu71] Peter Deussen. *Halbgruppen und Automaten.* Springer, 1971.

[DH91] Robert Dale and Nicholas Haddock. Content determination in the generation of referring expressions. *Computational Intelligence*, 7(4), 1991.

[DHRS92] Robert Dale, Eduard H. Hovy, Dietmar Rösner, and Oliviero Stock, editors. *Aspects of Automated Natural Language Generation.* LNAI 587. Springer, 1992.

[Die92] Scott Dietzen. *A Language for Higher-Order Explanation-Based Learning.* PhD thesis, School of Computer Science, Carnegie Mellon University, Jan. 1992.

[DMZ90] R. Dale, C. Mellish, and M. Zock, editors. *Current Research in Natural Language Generation.* Academic Press, London, 1990.

[EP93] Andrew Edgar and Francis Jeffry Pelletier. Natural language explanation of natural deduction proofs. In *Proc. of the first Conference of the Pacific Association for Computational Linguistics.* Centre for Systems Science, Simon Fraser University, 1993.

[FGT92] William M. Farmer, Joshua D. Guttman, and F. Javier Thayer. Little theories. In D. Kapur, editor, *Proc. of 11th International Conference on Automated Deduction*, pages 567–581. Springer, 1992.

[FGT93] William M. Farmer, Joshua D. Guttman, and F. Javier Thayer. IMPS: An interactive mathematical proof system. *Journal of Automated Reasoning*, 11:213–248, 1993.

[FHN72] R. R. Fikes, P. E. Hart, and N. J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3:251–288, 1972.

[FM90a] Steven Feiner and Kathleen McKeown. Coordinating text and graphics in explanation generation. In Tom Dietterich and William Swartont, editors, *Proc. of AAAI-90*, pages 442–449. MIT Press, 1990.

[FM90b] Steven Feiner and Kathleen McKeown. Generating coordinated multimedia explanations. In *Proc. CAIA-90 (6th IEEE Conf. on Artificial Intelligence applications)*, pages 290–296, March 1990.

[Gen35] Gerhard Gentzen. Untersuchungen über das logische Schließen I. *Math. Zeitschrift*, 39:176–210, 1935.

[GG84] D. Gabbay and F. Guenthner. *Handbook of Philosophical Logic*. D. Reidel Publishing, 1984.

[GJW83] B. J. Grosz, A. Joshi, and S. Weinstein. Providing a unified account of definite noun phrases in discourse. In *Proc. of the 21st Annual Meeting of the Association for Computational Linguistics*, pages 44–50, 1983.

[GN87] M. R. Genesereth and N. J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, 1987.

[Gri75a] H. P. Grice. Logic and conversation. In P. Cole and J. L. Morgan, editors, *Syntax and Semantics*, volume 3. Academic Press, 1975.

[Gri75b] Joseph E. Grimes. *The Thread of Discourse*. Mouton, The Hague, Paris, 1975.

[Gro77] Barbara. J. Grosz. *The Representation and Use of Focus in Dialogue Understanding*. PhD thesis, Univ. of California, Berkeley, 1977.

[GS86] Barbara J. Grosz and Candace L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204, 1986.

[Hay79] Patrick J. Hayes. The logic of frames. In *Readings in Knowledge Representation*, pages 287–294. Morgan Kaufmann, 1985, 1979.

[HHNT86] H. Holland, K. J. Holyoak, R. E. Nisbett, and P. R. Thagard. *Induction, Processes of Inference, Learning and Discovery.* MIT Press, 1986.

[Hin73] J. Hintikka. *Logic, Language-games, and Information.* Oxford UniversityPress, Oxford, 1973.

[HKK92a] Xiaorong Huang, Manfred Kerber, and Michael Kohlhase. Methods - the basic units for planning and verifying proofs. SEKI Report SR-92-20, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, Germany, 1992.

[HKK+92b] Xiaorong Huang, Manfred Kerber, Michael Kohlhase, Erica Melis, Daniel Nesmith, Jörn Richts, and Jörg Siekmann. Ω–MKRP – a proof development environment. SEKI Report SR-92-22, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, Germany, 1992. a shorter version presented at the Workshop on automated theorem proving, IJCAI-93, Chambéry, France.

[HKK94] Xiaorong Huang, Manfred Kerber, and Michael Kohlhase. Theorem proving as an interleaving process of planning and verification. SEKI Report to appear, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, Germany, 1994.

[Hov88] Eduard H. Hovy. *Generating Natural Language under Pragmatic Constraints.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1988.

[Hov90] Eduard H. Hovy. Pragmatics and natural language generation. *Artificial Intelligence*, 43:153–197, 1990.

[Hua89] Xiaorong Huang. Proof transformation towards human reasoning style. In Dieter Metzing, editor, *Proc. of 13 Annual German Conference on Artificial Intelligence*, number 216 in Informatik-Fachberichte, pages 37–42. Springer, 1989.

[Hua90] Xiaorong Huang. Reference choices in mathematical proofs. In L. C. Aiello, editor, *Proc. of 9th European Conference on Artificial Intelligence*, pages 720–725. Pitman Publishing, 1990.

[Hua91] Xiaorong Huang. An extensible natural calculus for argument presentation. SEKI-Report SR-91-03, Fachbereich Informatik, Universität Kaiserslautern, Kaiserslautern, Germany, 1991.

[Hua92] Xiaorong Huang. Applications of assertions as elementary tactics in proof planning. In V. Sgurev and B. du Boulay, editors, *Artificial Intelligence V - Methodology, Systems, Applications*, pages 25–34. Elsevier Science, the Netherlands, 1992.

[Hua93a] Xiaorong Huang. An explanatory framework for human theorem proving. In Hans Jürgen Ohlbach, editor, *GWAI-92: Advances in Artificial Intelligence*, LNAI 671, pages 55–66. Springer, 1993.

[Hua93b] Xiaorong Huang. A reconstructive approach towards proof presentation. In Jürgen Avenhaus and Jörg Denzinger, editors, *Proc. of the Annual Meeting of "GI-Fachgruppe 'Deduktionssysteme'"*, SEKI-Report SR-93-11, page 12, Kaiserslautern, Germany, 1993.

[Hut90] Dietter Hutter. Guiding induction proofs. In Mark E. Stickel, editor, *Proc. 10th International Conference on Automated Deduction*, pages 147–161. Springer, 1990.

[JL83] Philip N. Johnson-Laird. *Mental Models.* Harvard Univ. Press, Cambridge, Massachusetts, 1983.

[JLB90] Philip N. Johnson-Laird and Ruth. M.J Byrne. *Deduction.* Ablex Publishing, 1990.

[Kan77] R. N. Kantor. *The Menagement and Comprehension of Discourse Connection by Pronouns in English.* PhD thesis, Ohio State University, 1977.

[Kem87] G. Kempen, editor. *Natural Language Generation.* Nijhoff, Dordrecht, 1987.

[Ker91] Manfred Kerber. On the representation of mathematical knowledge in frames and its consistency. In Michel De Glas and Dov Gabbay, editors, *Proc. WOCFAI '91*, pages 293–301, Paris, France, 1991. Angkor.

[KF93] Anne Kilger and Wolfgang Finkler. TAG-based incremental generation. DFKI Report forthcoming, German Research Center for Artificial Intelligence, Saarbrücken, Germany, 1993.

[Kil94] Anne Kilger. Using UTAGs for incremental and parallel generation. *Computational Intelligence*, forthcoming, 1994.

[Kun77] S. Kuno. Generative discourse analysis in America. In J. Petofi, editor, *Cureent Trends in Textlinguistics.* De Gruyter, 1977.

[Lak70] George Lakoff. Linguistics and natural logic. *Syntheses*, 22:151–271, 1970.

[Lak87] George Lakoff. *Women, Fire, and Dangerous Things, What Categories Reveal about the Mind.* The University of Chicago Press, 1987.

[Lev89] Willem J. M. Levelt. *Speaking: From Intention to Articulation.* ACL-MIT Press Series in Natural Language Processing. MIT Press, 1989.

[LG90] Douglas B. Lenat and R. V. Guha. *Building Large Knowledge-Based Systems, Representation and Inference in the Cyc Project.* Addison-Wesley, 1990.

[Lin89] Christoph Lingenfelder. Structuring computer generated proofs. In N.S. Sridharan, editor, *Proc. of IJCAI-89*, pages 378–383. Morgan Kaufmann, 1989.

[Lin90] Christoph Lingenfelder. *Transformation and Structuring of Computer Generated Proofs*. PhD thesis, Universität Kaiserslautern, Kaiserslautern, Germany, 1990.

[LN77] Peter H. Lindsay and Donald A. Norman. *An Introduction to Psychology*. Academic Press, 1977.

[McC87] Kathleen F. McCoy. Contextual effects on responses to misconceptions. In Gerard Kempen, editor, *Natural Language Generation*, pages 43–54. Martinus Nijhoff, 1987.

[McC90] William McCune. Otter 2.0. In Mark E. Stickel, editor, *Proc. of 10th International Conference on Automated Deduction*, pages 663–664. Springer, 1990.

[McD83] David D. McDonald. Natural language generation as a computational problem. In *Brady/Berwick: Computational Models of Discourse*. MIT Press, 1983.

[McK85] Kathleen R. McKeown. *Text Generation*. Cambridge University Press, Cambridge, UK., 1985.

[Mel93] Erica Melis. Change of representation in theorem proving by analogy. SEKI-Report SR-93-07, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, Germany, 1993.

[Mil83] Dale A. Miller. *Proofs in Higher-Order Logic*. PhD thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, 1983.

[MKKC86] Tom M. Mitchell, Richard M. Keller, and Smadar T. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1:47–80, 1986.

[MMN90] K.R. McKeown, J.D. Moore, and S. Nirenburg, editors. *Proceedings of the 5th International Workshop on Natural Language Generation*, Dawson, PA., 1990.

[Moo89] Johanna Doris Moore. *A Reactive Approach to Explanation in Expert and Advice-Giving Systems*. PhD thesis, Univ. of California, Los Angeles, California, USA, 1989.

[Moo90] Raymond J. Mooney. Learning plan schemata from observation: Explanantion-based learning for plan recognition. *Cognitive Science*, 14:483–509, 1990.

[MT87] William C. Mann and Sandra A. Thompson. Rhetorical structure theory: A theory of text organization. Technical Report ISI/RS-87-190, USC Information Sciences Institute, 1987.

[Nes92] Dan Nesmit(Editor). KEIM-Manual. Version 0, 1992. Universisät des Saarlandes, Saarbücken, Germany.

[New90] Allen Newell. *Unified Theories in Cognition.* Harvard University Press, Cambridge, MA, 1990.

[NR81] A. Newell and P.S. Rosenbloom. Mechanism of skill acquisition and the law of practice. In J. R. Anderson, editor, *Cognitive Skills and Their Acquisition.* Hillsdale, NJ: Erlbaum, 1981.

[NRL89] Allen Newell, Paul S. Rosenbloom, and John E. Laird. Symbolic architectures for cognition. In Michael I. Posner, editor, *Foundations of Cognitive Science.* MIT Press, 1989.

[Och79] Elinor Ochs. Planned and unplanned discourse. *Syntax and Semantics*, 12:51–80, 1979.

[OS91] Hans Jürgen Ohlbach and Jörg H. Siekmann. The Markgraf Karl Refutation Procedure. In Jean-Louis Lassez and Gordon Plotkin, editors, *Computational Logic - Essays in Honor of Alan Robinson*, chapter Chapter 2, pages pages 41–112. MIT Press, 1991.

[Par88] Cecile Paris. Tailoring object descriptions to a user's level of expertise. *Computational Linguistics*, 14:64–78, 1988.

[Pas93] Dominique Pastre. Automated theorem proving in mathematics. *Annals on Artificial Intelligence and Mathematics*, 8(3-4):425–447, 1993.

[Pat93] T. Pattabhiraman. *Aspects of Salience in Natural Language Generation.* PhD thesis, School of Computer Science, Simon Fraser University, Canada, 1993.

[PC93] T. Pattabhiraman and Nick Cercone. Decision-theoretic salience interactions in language generation. In Ruzena Bajcsy, editor, *Proc. of IJCAI-93*, volume 2, pages 1246–1252, Chambéry, France, 1993. Morgan Kaufmann.

[Pfe87] Frank Pfenning. *Proof Transformation in Higher-Order Logic.* PhD thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, 1987.

[PN90] Frank Pfenning and Daniel Nesmith. Presenting intuitive deductions via symmetric simplification. In Mark E. Stickel, editor, *Proc. of 10th International Conference on Automated Deduction*, LNAI 449, pages 336–350. Springer, 1990.

[Pól45] George Pólya. *How to Solve it.* Princeton Univ. Press, 1945.

[Pol92] Thad A. Polk. *Verbal Reasoning.* PhD thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, 1992.

[Pos89] Michael I. Posner(ED.). *Foundations of Cognitive Science.* MIT Press, 1989.

[Pri67] A. Prior. *Past, Present and Future.* Oxford University Press, 1967.

[PSM91] C.L. Paris, W.R. Swartout, and W.C. Mann, editors. *Natural Language Generation in Artificial Intelligence and Computational Linguistics.* Kluwer, Norwell, MA., 1991.

[RD92] Ehud Reiter and Robert Dale. A fast algorithm for the generation of referring expressions. In *Proc. of COLING-92*, volume 1, pages 232–238, 1992.

[Rei85] Rachel Reichman. *Getting Computer to Talk Like You and Me. Discourse Context, Focus, and Semantics.* MIT Press, 1985.

[Rei91] Norbert Reithinger. *Eine parallele Architektur zur inkrementeller Dialogbeiträge.* PhD thesis, Universität des Saarlandes, 1991. Also available as book, Infix, Sankt Augustin, 1991.

[Rip83] Lance J. Rips. Cognitive processes in propositional reasoning. *Psychological Review*, 90:38–71, 1983.

[RL86] Paul S. Rosenbloom and John E. Laird. Mapping explanation-based generalization onto soar. In *Proc. of AAAI-86*, pages 561–567, 1986.

[RU71] N. Rescher and A. Urquhart. *Temporal Logic.* Springer, 1971.

[SD93] Inger Sonntag and Jörg Denzinger. Extending automatic theorem proving by planning. SEKI-Report SR-93-02 (SFB), Fachbereich Informatik, Universität Kaiserslautern, Kaiserslautern, Germany, 1993.

[SGBM75] R. L. Smith, W. H. Graves, L. H. Blaine, and V. G. Marinov. Computer-assisted axiomatic mathematics: Informal rigor. In O. Lecarme and R. Lewis, editors, *Computers in Education*, pages 803–809. North-Holland, 1975.

[She26] H. R. Shepherd. *The fine art of writing.* Macmillan, New York, 1926.

[Sho76] E. H. Shortliffe. *Computer-Based Medical Consultations: MYCIN.* Elsevier, New York, 1976.

[Sib90] Penelope Sibun. The local organization of text. In K.R. McKeown, J.D. Moore, and S. Nirenburg, editors, *Proc. of the fifth international natural language generation workshop*, pages 120–127, Dawson, Pennsylvania, 1990.

[Sid79] Candace L. Sidner. *Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse.* PhD thesis, Massachusetts Institute of Technology, 1979.

[SK93] Barbara Schütt and Manfred Kerber. A mathematical knowledge base for proving theorems in semigroup and automata theory – part i –. SEKI Working Papers SWP-93-02, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, Germany, Germany, 1993.

[SM77] Donald F. Stanat and David F. McAllister. *Discrete Mathematics in Computer Science.* Prentice-Hall, Englewood Cliffs, N. J., 1977.

[Smu78] Raymond M. Smullyan. *What is the Name of this Book.* Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

[SP88] Remko Scha and Livia Polani. An augmented context free grammar for discourse. In *Proc. COLING-88*, pages 573–577, 1988.

[SS89] Manfred Schmidt-Schauß. *Computational Aspects of an Order-Sorted Logic with Term Declarations.* LNAI 395. Springer, 1989.

[Sti86] Mark E. Stickel. Schubert's steamroller problem: Formulations and solutions. *Journal of Automated Reasoning*, 2(1), 1986.

[Van89] Kurt VanLehn. Problem solving and cognitive skill acquisition. In Michael I. Posner, editor, *Foundations of Cognitive Science*, chapter 14. MIT Press, 1989.

[WAB+92] Wolfgang Wahlster, Elisabeth André, Som Bandyopadhyay, Winfried Graf, and Thomas Rist. WIP: The coordinated generation of multimodal presentations from a common representation. In A. Ortony, J. Slack, and O. Stock, editors, *Computational Theories of Communication and their Applications.* Springer, 1992.

[WAF+93] Wolfgang Wahlster, Elisabeth André, Wolfgang Finkler, Hans-Jürgen Profitlich, and Thomas Rist. Plan-based integration of natural language and graphics generation. *Artificial Intelligence*, 63:387–428, 1993.

[Wal87] Christoph Walther. *A Classification of Many-sorted Calculus Based on Resolution and Paramodulation.* Pitman & Kaufman, 1987.

[Web89] Bonnie L. Webber. Deictic reference and discourse structure. Technical Report MS-CIS-89-55, Dept. of Computer and Information Science, Univ. of Pennsylvania, 1989.

[Wei93] Christoph Weidenbach. Extending the resolution method with sorts. In Ruzena Bajcsy, editor, *Proc. of IJCAI-93*, volume 1, pages 60–65, Chambéry, France, 1993. Morgan Kaufmann.

[WJL72] P. C. Wason and P. N. Johnson-Laird. *Psychology Reasoning. Structure and Content.* Batsford, London, 1972.

[WS89] M. R. Wick and J. R. Slagle. An explanation facility for today's expert systems. *IEEE Expert*, 4(1):26–36, 1989.

[WT92] Michael R. Wick and William B. Thompson. Reconstructive expert system explanation. *Artificial Intelligence*, 54:33–70, 1992.

[ZS88] M. Zock and G. Sabah, editors. *Advances in Natural Language Generation.* Pinter, London, 1988.

[Zuk86] Ingrid Zukerman. *Computer Generation of Meta-Technical Utterances in Tutoring Mathematics.* PhD thesis, University of Calaifornia, Los Angeles, 1986.