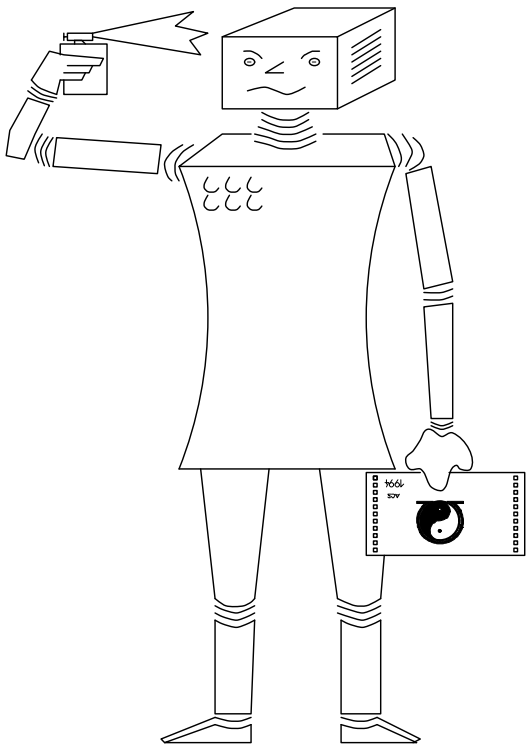

Omega Publication

Proof Verbalization as an Application of NLG

Xiaorong Huang and Armin Fiedler

Published as: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence

Proof Verbalization as an Application of NLG

Xiaorong Huang* **Armin Fiedler**
Fachbereich Informatik, Universität des Saarlandes
Postfach 15 11 50, D-66041 Saarbrücken, Germany
{huang|afiedler}@cs.uni-sb.de

Abstract

This paper describes the linguistic part of a system called *PROVERB*, which transforms, abstracts, and verbalizes machine-found proofs into formated texts. Linguistically, the architecture of *PROVERB* follows most application oriented systems, and is a pipe-lined control of three components. Its macroplanner linearizes a proof and plans mediating communicative acts by employing a combination of hierarchical planning and focus-guided navigation. The microplanner then maps communicative acts and domain concepts into linguistic resources, paraphrases and aggregates such resources to produce the final Text Structure. A Text Structure contains all necessary syntactic information, and can be executed by our realizer into grammatical sentences. The system works fully automatically and performs particularly well for textbook size examples.

1 Introduction

PROVERB is a text planner that verbalizes natural deduction (ND) style proofs [Gentzen, 1935; Huang, 1994b]. Several similar attempts can be found in previous works. The system EXPOUND [Chester, 1976] is an example of *direct translation*: Although a sophisticated linearization is applied on the input ND proofs, the steps are then translated locally in a template driven way. ND proofs were tested as inputs to an early version of MUMBLE [McDonald, 1983], the main aim however, was to show the feasibility of the architecture. A more recent attempt can be found in THINKER [Edgar and Pelletier, 1993], which implements several interesting but isolated proof presentation strategies. *PROVERB* therefore can be seen as the first serious attempt to build a comprehensive system that produces adequate argumentative texts from ND style proofs.

Although a multitude of architectures have been proposed for NLG systems, *PROVERB* employs a pipe line architecture consisting of three parts, like most application-oriented systems. The architecture of *PROVERB* is illustrated in Fig. 1[1].

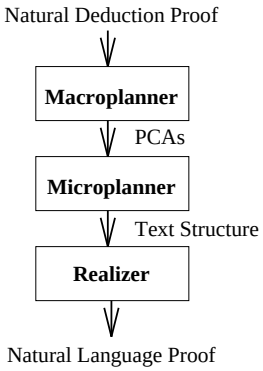

Figure 1: Architecture of *PROVERB*

The *macroplanner* of *PROVERB* accepts as input a natural deduction style proof, and produces *proof communicative acts* which are structured into hierarchical attentional spaces. To do so, it uses a strategy which combines *hierarchical planning* and *focus-guided navigation*.

More detailed linguistic decisions are made by the *microplanner*. It makes reference choices, chooses between linguistic resources for domain concepts, combines and reorganizes such resources into paragraphs and sentences. As the representation which supports all these operations, the microplanner of *PROVERB* adopts a variation of Meteer's Text Structure, which is also its output.

Our realizer, TAG-GEN, is a syntactic generator based on the grammar formalism TAG [Kilger and Finkler, 1995].

Section 2 and Section 3 are devoted to the macroplanner and the microplanner, respectively. Section 4 contains a complete example. Finally, we shall conclude this paper with a discussion in Section 5.

*Xiaorong Huang's current email address is: xh@FormalSys.ca

[1] In the field of NLG the first two components are called "planners," since they make decisions that will be executed by the realization component. Many approaches differ significantly from general definitions of planner in AI. In particular, approaches towards microplanning often resemble more rule-based systems.

2 Macroplanning: Hierarchical Planning and Focus-Guided Navigation

Most current text planners adopt a hierarchical planning approach [Hovy, 1988; Moore and Paris, 1989; Dale, 1992; Reithinger, 1991]. Nevertheless there is psychological evidence that language has an unplanned, spontaneous aspect as well [Ochs, 1979]. Based on this observation, Sibun [Sibun, 1990] implemented a system for generating descriptions of objects with a strong domain structure, such as houses, ships and families. While a hierarchical planner recursively breaks generation tasks into subtasks, local organization navigates the domain object following the local focus of attention.

PROVERB combines both of these approaches within a uniform planning framework [Huang, 1994a]. The *hierarchical planning* splits the task of presenting a particular proof into subtasks of presenting subproofs. While the overall planning mechanism follows the RST-based planning approach [Hovy, 1988; Moore and Paris, 1989; Reithinger, 1991], the planning operators more resemble the schemata in schema-based planning [McKeown, 1985; Paris, 1988]. *Local navigation* operators simulate the unplanned aspect, where the next conclusion to be presented is chosen under the guidance of a local focus mechanism.

The two kinds of *planning operators* are treated differently. Since hierarchical planning operators embody explicit communicative norms, they are given a higher priority. Only when none of them is applicable will a local navigation operator be chosen.

2.1 Proof Communicative Acts

Proof communicative acts (PCAs) are the primitive actions planned by the macroplanner of *PROVERB*. Like speech acts, they can be defined in terms of the communicative goals they fulfill as well as their possible verbalizations. An example of a simplistic one conveying the derivation of a new intermediate conclusion is the PCA

```
(Derive Reasons:  (a ∈F, F ⊆G)
        Method: def-subset
        Conclusion:  a ∈G)
```

Depending on the reference choices, the following is a possible verbalization:

> "Since a is an element of F and F is a subset of G, a is an element of G by the definition of subset."

There are also PCAs that predicate actions planned for further presentation and thereby update the global attentional structure. For instance, the PCA

```
(Begin-Cases Goal:   Formula
             Assumptions:  (A B))
```

creates two attentional spaces with A and B as the assumptions, and *Formula* as the goal by producing the verbalization:

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

$$\frac{\cdots}{\vdots}\ ,\ \frac{F}{\vdots}\ ,\ \frac{G}{\vdots}$$
$$\frac{?L_4 : F \vee G \qquad ?L_2 : Q \qquad ?L_3 : Q}{?L_1 : \triangle \vdash Q}\ \text{CASE}$$

Figure 2: Proof Schema Case

2.2 Hierarchical Planning

Hierarchical planning operators represent communicative norms concerning how the task of presenting a proof can be split into subtasks of presenting subproofs, and how the subproofs can be mapped onto some linear order. Let us look at one such operator, which handles the goal of presenting a proof by case analysis. The corresponding schema of such a proof tree is shown in Fig. 2, where the subproof rooted by $?L_4$ leads to $F \vee G$, while subproofs rooted by $?L_2$ and $?L_3$ are the two cases proving Q by assuming F or G, respectively. The applicability encodes the two scenarios of case analysis, where we do not go into details. In both circumstances this operator first presents the part leading to $F \vee G$, and then proceeds with the two cases. It also inserts certain PCAs to mediate between parts of proofs. This procedure is captured by the planning operator below (note that the verbalizations given are only one possible paraphrase):

Case-Implicit

- Precondition: ((task $?L_1$) $\vee$ (local-focus $?L_4$)) $\wedge$ (not-conveyed ($?L_2$ $?L_3$))
- Acts:
  1. if $?L_4$ has not been conveyed, then present $?L_4$ (subgoal 1)
  2. a PCA with a verbalization: "First, let us consider the first case by assuming F."
  3. present $?L_2$ (subgoal 2)
  4. a PCA with a verbalization: "Next, we consider the second case by assuming G."
  5. present $?L_3$ (subgoal 3)
  6. mark $?L_1$ as conveyed
- features: (hierarchical-planning compulsory implicit)

The slot features indicates that this is a higher priority operator (compulsory) and should be chosen when a more implicit style is preferred by the user.

2.3 Planning as Navigation

The *local navigation* operators simulate the unplanned part of proof presentation. Instead of splitting presentation goals into subgoals, they follow the local derivation relation to find a proof step to be presented next.

The Local Focus The node to be presented next is suggested by the mechanism of *local focus*. In *PROVERB*, our local focus is the last derived step, while focal centers are semantic objects mentioned in the local focus. Although logically any proof node which uses the local focus as a premise could be chosen for the next step, usually the one with the greatest semantic overlap with the *focal centers* is preferred. In other words, if one has proved a property about some semantic objects, one will tend to continue to talk about these particular objects, before turning to new objects. Let us examine the situation when the proof below is awaiting presentation.

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

Assume that node [1] is the local focus, $\{a, b\}$ is the set of focal centers, [3] is a previously presented node and node [5] is the current task. [2] is chosen as the next node to be presented, since it does not (re)introduce any new semantic objects and its overlap with the focal centers ($\{a, b\}$) is larger than the overlap of [4] with the focal centers ($\{b\}$). Due to space restrictions, no navigation operators are discussed in detail.

3 Microplanning: Choosing and Organizing Linguistic Resources

Many of the first NLG systems link their information structure to the corresponding linguistic resources either through predefined templates or via careful engineering for a specific application. Therefore their expressive power is restricted (see [Meteer, 1992] for an extensive discussion). First experiments with *PROVERB* using a simplistic microplanning mechanism resulted in very mechanical texts. According to our analysis, there are at least two linguistic phenomena that call for appropriate microplanning techniques.

3.1 Why is Microplanning Needed?

First, naturally occurring proofs contain paraphrases with respect to rhetorical relations, as well as to logical functions or predicates. For instance, the derivation of B from A can be verbalized as:

"Since A, B." or as "A leads to B."

The logic predicate $para(C1, C2)$, also, can be verbalized as:

"Line $C1$ parallels line $C2$." or as
"The parallelism of the lines $C1$ and $C2$."

Second, without microplanning *PROVERB* generates text structured mirroring the information structure of the proof and the formulae. This means that every step of derivation included by the macroplanner is translated into a separate sentence, and formulae are recursively verbalized. As an instance of the latter, the formula

$$Set(F) \wedge Subset(F, G) \qquad (1)$$

is verbalized as

"F is a set. F is a subset of G."

although the following is much more natural:

"The set F is a subset of G."

Therefore, we came to the conclusion that an intermediate level of representation is necessary that allows for flexible combinations of linguistic resources. In Section 3.2 we describe how Meteer's Text Structure can be adopted as our central representation. Sections 3.3 and 3.4 are devoted to paraphrases and aggregation rules, two of the major tasks of our microplanner.

3.2 Text Structure in *PROVERB*

Text Structure was first proposed by Meteer [Meteer, 1992] in order to bridge the generation gap between the representation in the application program and the linguistic resources provided by the language. Meteer's Text Structure is organized as a tree, in which each node represents a constituent of the text and is typed in terms of semantic categories.

The main role of the *semantic categories* is to provide vocabularies which specify type restrictions for nodes. They define how separate Text Structures can be combined, and ensure that the planner only build expressible Text Structures. For instance, if tree A should be expanded at node n by tree B, the resulting type of B must be compatible to the type restriction attached to n. Following Panaget [Panaget, 1994], however, we split the type restrictions into two orthogonal dimensions: the ideational dimension in terms of the *Upper Model* [Bateman *et al.*, 1990], and the *hierarchy of textual semantic categories* to be discussed below. Technically speaking, the Text Structure in *PROVERB* is a tree recursively composed of kernel subtrees or composite subtrees:

An atomic *kernel subtree* has a head at the root and arguments as children, representing basically a predicate/argument structure.

Composite subtrees can be divided into two subtypes: the first has a special *matrix* child and zero or more *adjunct* children and represents linguistic hypotaxis, the second has two or more *coordinated* children and stands for parataxis.

Each node is typed both in terms of the Upper Model and the hierarchy of textual semantic categories. The Upper Model is a domain-independent property inheritance network of concepts that are hierarchically organized according to how they can be linguistically expressed. Fig. 3 shows a fragment of the Upper Model in *PROVERB*. For every domain of application, domain-specific concepts must be identified and placed as an extension of the Upper Model.

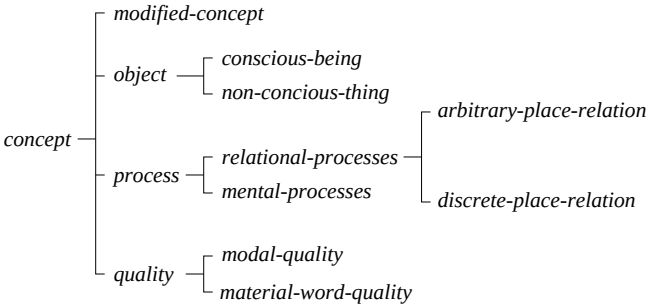

Figure 3: A Fragment of the Upper Model in *PROVERB*

The hierarchy of textual semantic categories is also a domain-independent property inheritance network. The concepts are organized in a hierarchy based on their textual realization. For example, the concept clause-modifier-rankingI is realized as an adverb, clause-modifier-rankingII as a prepositional phrase, and clause-modifier-embedded as an adverbial clause. Fig. 4 shows a fragment of the hierarchy of textual semantic categories.

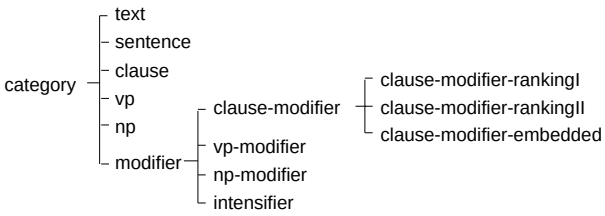

Figure 4: A Fragment of the Hierarchy of Textual Semantic Categories in *PROVERB*

3.3 Paraphrasing in *PROVERB*

The mapping from the content to the linguistic resources now happens in a two-staged way. While Meteer associates the application program objects (APOs) directly with so-called *resources trees*, we map APOs into Upper Model objects, which in turn are expanded to the Text Structures. A practical advantage of this two-stadged process is worth noting. Instead of having to construct resource trees for APOs, the user of our system only needs to define a mapping from the APOs to Upper Model objects (UMOs).

When mapping APOs to UMOs, the microplanner must choose among available alternatives. For example, the application program object `para`, that stands for the logical predicate denoting the parallelism relation between lines, may map in five different Upper Model concepts. In the 0-place case, `para` can be mapped into *object* leading to the noun "parallelism," or *quality*, leading to the adjective "parallel." In the binary case, the choices are *property-ascription* that may be verbalized as "x and y are parallel," *quality-relation* that allows the verbalization as "x is parallel to y", or *process-relation*, that is the formula "$x \parallel y$."

The mapping of Upper Model objects into the Text Structure is defined by so-called *resource trees*, i.e. reified instances of Text Structure subtrees. The alternative resource trees of an Upper Model concept are assembled in its *realization class*.

With the help of a concrete example we shall illustrate how the Text Structure generator chooses among paraphrases and avoids building inexpressible Text Structures via type checking.

Example We examine a simple APO *derive*(`para(C1,C2)`,B). Note that B stands for a conclusion which will not be examined here.

In the current implementation, the rhetorical relation *derive* is only connected to one Upper Model concept *derive*, a subconcept of *cause-relation*. The realization class associated to the concept, however, contains several alternative resource trees. The verbalization of two variations is listed below:

- B, since A.
- Because of A, B.

The resource tree of the first alternative is given in Fig. 5.

The logic predicate `para(C1, C2)` can be mapped to one of the following Upper Model concepts, where we always include one possible verbalization:

- *quality-relation*(`para`, `C1`, `C2`)
  (line $C1$ is parallel to $C2$)
- *process-relation*(`para`, `C1`, `C2`)
  ($C1 \| C2$)

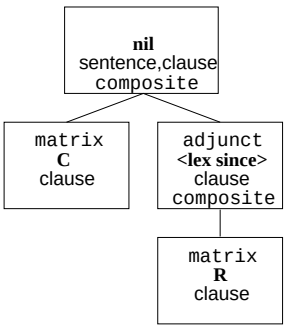

Figure 5: A Resource Tree for *derive* with Reason R and Conclusion C

- *property-ascription*(`para`, `C1` $\wedge$ `C2`)
  (lines $C1$ and $C2$ are parallel)

The *property-ascription* version, in turn, can be realized in two forms, represented by the two resource trees in Fig. 6.

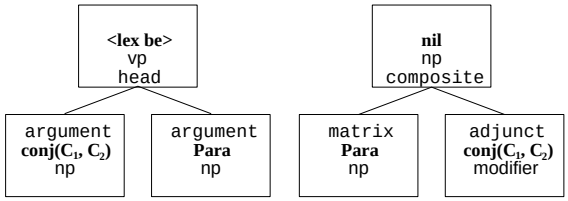

Figure 6: Textual Variations in form of Resource Trees

Type checking must ensure that the constructed Text Structure be compatible along both the ideational and the textual dimension. In this example, the combination of the tree in Fig. 5 and the first tree in Fig. 6 is compatible and will lead to the verbalization:

"B, since $C1$ and $C2$ are parallel."

The second tree in Fig. 6, however, can only be combined with another realization of *derive*, resulting in:

"Because of the parallelism of line $C1$ and line $C2$, B."

In our current system we concentrate on the mechanism and are therefore still experimenting with heuristics that control the choice of paraphrases. One interesting rule is to distinguish between general rhetorical relations and domain specific mathematical concepts. While the former should be paraphrased to increase the flexibility, consistency of the latter helps the user to identify technical concepts.

3.4 Aggregation

Although paraphrase generation already increases the flexibility in the text, the default verbalization strategy will still expand the Text Structure by recursively descending the proof and formula structure, and thereby produces linguistic structures isomorphic to that of a formula. To achieve the second verbalization of equation (1) in Section 3.1, however, we have to combine $Set(F)$ and $Subset(F, G)$ to form an embedded structure $Subset(Set(F), G)$. This textual operation eliminates one of the duplicates of F. We call it *aggregation*.

Aggregation rules operate on APOs, which, before mapped to UMOs, can be viewed as variables for UMOs (for convenience, we continue to refer to them as APOs). For instance, the embedded structure $Subset(Set(F), G)$ documents a textual decision that no matter how *Subset* and *Set* are instantiated, the argument F in $Subset(F, G)$ will be replaced by $Set(F)$. In this sense, our rules work with such variables at the *semantic level* of the Upper Model. So far, we have investigated three types of aggregation, as illustrated in Fig. 7.

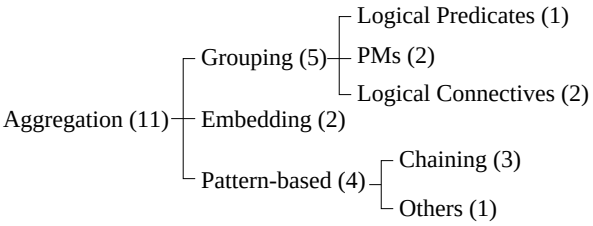

Figure 7: A Classification of Aggregation Rules in *PROVERB* and their Numbers

Semantic Grouping We use semantic grouping to characterize the merge of two parallel Text Structure objects with the same top-concept by grouping their arguments. Two APOs are parallel in the sense that they have the same parent node. The general form of this type of rules can be characterized by the pattern as given below:

Rule Pattern A

$$\frac{P[a] + P[b]}{P[a \bigoplus b]}$$

The syntax of our rules means that a Text Structure of the form above the bar will be transformed into one of the form below the bar. Viewing Text Structure as a tree, $P[a]$ and $P[b]$ are both sons of $+$, they are merged together by grouping the arguments a and b under another operator $\bigoplus$. In the first rule below, $+$ and $\bigoplus$ are identical.

Rule A.1 (Predicate Grouping)

$$\frac{P[a] + P[b]}{P[a + b]}$$

where $+$ can be either a logical $\wedge$ or a logical $\vee$, and P stands for a logical predicate. The following example illustrates the effect of this rule.

$Set(F) \wedge Set(G)$

"F is a set. G is a set."

are aggregated to:

$Set(F \wedge G)$

"F and G are sets."

The rule covers the predicate grouping rule reported in [Dalianis and Hovy, 1993]. This is also the best place to explain why we apply aggregation before choosing concrete linguistic resources. If the two occurrences of *Set* are instantiated differently, this rule will be blocked.

By instantiating $+$, $\bigoplus$ and P in pattern A to different logical connectives and derivation relations, we have altogether five rules in this category. The correctness of the rules in this category with respect to the information conveyed is guaranteed by the semantics of the corresponding Upper Model.

Semantic Embedding The next category of aggregation rules handles parallel structures which are not identical. In this case, some of them may be converted to embedded structures, as is done by the following rule.

Rule B.1 (Object Embedding)

$$\frac{P[T] \wedge Q[T]}{Q[P[T]]}$$

We will not discuss the preconditions of this rule in detail, but only illustrate its effect by an example. Getting as input two APOs in a conjunction that contain a common APO F:

$Set(F) \wedge Subset(F, G)$

"F is a set and F is a subset of G."

rule B.1 aggregates them by removing one redundant F:

$Subset(Set(F), G)$

"The set F is a subset of G."

Actually, for mathematical texts we have only used two embedding rules, the other being the dual of rule B.1 where P and Q change their places.

Pattern-based Optimization rules Rules in the third category involve more complex changes of the textual structure. They could be understood as some domain-specific communicative conventions, and must be explored in every domain of application. See [Fiedler, 1996] for details.

4 A Complete Example

In this section, we present a short example of *PROVERB*'s output. The input is a machine-found proof for a theorem taken from a mathematical textbook. A user may choose the style of the output text by tuning two parameters: implicit vs. explicit, abstract vs. detailed. Given abstract and explicit as options, the macroplanner generates the following sequence of PCAs:

```
(THEOREM Reasons (group(F,*), subgroup(U,F,*),
                  unit(F,1,*), unit(U,1_U,*)),
         Conclusion 1_U = 1)
(PROOF)
(ATTENTION-BEGIN)
(ASSUME Conclusion group(F,*))
(ASSUME Conclusion subgroup(U,F,*))
(ASSUME Conclusion unit(F,1,*))
(ASSUME Conclusion unit(U,1_U,*))
(ATTENTION-BEGIN)
(ATTENTION-BEGIN)
(DERIVE Reasons unit(U,1_U,*), Conclusion 1_U ∈U)
(DERIVE Conclusion ∃x.x ∈U)
(ATTENTION-END)
(BEGIN-ASSUME-CHOICE-1 Conclusion u_1 ∈U,
                       Parameters u_1, x)
(ATTENTION-BEGIN)
...
(DERIVE Reasons group(F,*),
        Conclusion semigroup(F,*))
...
(DERIVE Conclusion 1_U = 1)
(ATTENTION-END)
(END-ASSUME-CHOICE-1 Conclusion 1_U = 1,
                     Parameters u_1)
(ATTENTION-END)
(ATTENTION-END)
(QED)
```

Note that the PCAs are segmented into a hierarchy of attentional spaces, which is used to make reference de-

cisions [Huang, 1997]. The microplanner maps this sequence of PCAs via Upper model objects into a Text Structure, which is used as linguistic specification of the text. It also aggregates Upper Model objects and paraphrases the concepts by combining various resource trees. Finally, our realization component TAG-GEN produces the following text:

Theorem:
Let F be a group, let U be a subgroup of F, and let 1 and 1_U be unit elements of F and U. Then 1_U equals 1.
Proof:
Let F be a group, let U be a subgroup of F, and let 1 and 1_U be unit elements of F and U.
Because 1_U is an unit element of U, $1_U \in U$. Therefore, there is x such that $x \in U$.
Let u_1 be such an x. Since $u_1 \in U$ and 1_U is an unit element of U, $u_1 * 1_U = u_1$. Since F is a group, F is a semigroup. Since U is a subgroup of F, $U \subset F$. Because $U \subset F$ and $1_U \in U$, $1_U \in F$. Similarly, because $u_1 \in U$ and $U \subset F$, $u_1 \in F$. Then, 1_U is a solution of $u_1 * x = u_1$.
Because $u_1 \in F$ and 1 is an unit element of F, $u_1 * 1 = u_1$. Since 1 is an unit element of F, $1 \in F$. Then, 1 is a solution of $u_1 * x = u_1$.
Therefore, 1_U equals 1. This conclusion is independent of the choice of u_1. ∎

Please note the variation in the text, in the structure of the sentences, and in using mathematical symbols vs. words. Moreover aggregation techniques reduced redundancies as in the sentence "let 1 and 1_U be unit elements of F and U."

5 Outlook

This paper examines *PROVERB* as an integration of sophisticated linguistic technologies for a concrete application. The system works particularly well with textbook size examples and runs fully automatically for every new example. The output texts are close to detailed proofs in textbooks and are basically accepted by the community of automated reasoning. To benefit from the microplaning techniques which significantly improve the fluency of text, however, linguistic resources must be introduced with each new domain of application. We are working on an interface to simplify this process.

Although developed for a specific application, we believe the main rationales behind of our system architecture are useful for natural language generation in general. The combination of hierarchical planning with focus-guided navigation provides an effective way of factoring out domain-dependent presentation knowledge from more general NLG techniques. While the macroplanning operators are designed for this specific domain, the framework and the rules of our microplanner represents domain-independent techniques.

With the increase of the number of examples tested, some of them over several pages, our experience already suggests some immediate adjustment and improvement of the techniques and strategies, in particular concerning the linearization in macroplanning, heuristic threshold values for discourse segmentation and reference choices, and the treatment of articles.

References

[Bateman *et al.*, 1990] J. A. Bateman, R. T. Kasper, J. D. Moore, and R. A. Whitney. A general organization of knowledge for natural language processing: the Penman upper model. Technical report, USC/Information Science Institute, 1990.

[Chester, 1976] D. Chester. The translation of formal proofs into English. *AI*, 7:178–216, 1976.

[Dale, 1992] R. Dale. *Generating Referring Expressions.* ACL-MIT Press Series in Natural Language Processing. MIT Press, 1992.

[Dalianis and Hovy, 1993] H. Dalianis and E. H. Hovy. Aggregation in natural language generation. In M. Zock, G. Adorni, and G. Ferrari, eds., *Proc. of 4th European Workshop on Natural Language Generation*, pages 67–73, 1993.

[Edgar and Pelletier, 1993] A. Edgar and F. J. Pelletier. Natural language explanation of natural deduction proofs. In *Proc. of 1st Conference of the Pacific Association for Computational Linguistics*, Vancouver, Canada, 1993.

[Fiedler, 1996] A. Fiedler. Mikroplanungstechniken zur Präsentation mathematischer Beweise. Master's thesis, Computer Science Department, Universität des Saarlandes, Saarbrücken, Germany, 1996.

[Gentzen, 1935] G. Gentzen. Untersuchungen über das logische Schließen I. *Mathematische Zeitschrift*, 39:176–210, 1935.

[Hovy, 1988] E. H. Hovy. *Generating Natural Language under Pragmatic Constraints.* Lawrence Erlbaum, Hillsdale, New Jersey, 1988.

[Huang, 1994a] X. Huang. Planning argumentative texts. In *Proc. of 15th International Conference on Computational Linguistics*, pages 329–333, Kyoto, Japan, 1994.

[Huang, 1994b] X. Huang. Reconstructing proofs at the assertion level. In A. Bundy, ed., *Proc. of 12th Conference on Automated Deduction*, number 814 in LNAI, pages 738–752, 1994. Springer Verlag.

[Huang, 1997] X. Huang. Planning reference choices for argumentative texts. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, 1997. forthcoming.

[Kilger and Finkler, 1995] A. Kilger and W. Finkler. Incremental generation for real–time applications. Research Report RR-95-11, DFKI, Saarbrücken, Germany, July 1995.

[McDonald, 1983] D. D. McDonald. Natural language generation as a computational problem. In Brady and Berwick, eds., *Computational Models of Discourse.* 1983.

[McKeown, 1985] K. R. McKeown. *Text Generation.* Cambridge University Press, Cambridge, United Kingdom, 1985.

[Meteer, 1992] M. W. Meteer. *Expressibility and the Problem of Efficient Text Planning.* Pinter Publishes, London, 1992.

[Moore and Paris, 1989] J. D. Moore and C. L. Paris. Planning text for advisory dialogues. In *Proc. of 27th Annual Meeting of the Association for Computational Linguistics*, pages 203–211, Vancouver, British Columbia, 1989.

[Ochs, 1979] E. Ochs. Planned and unplanned discourse. *Syntax and Semantics*, 12:51–80, 1979.

[Panaget, 1994] F. Panaget. Using a textual representational level component in the context of discourse or dialogue generation. In *Proc. of 7th International Workshop on Natural Language Generation*, pages 127–136, Kennebunkport, Maine, USA, 1994.

[Paris, 1988] C. Paris. Tailoring objects descriptions to a user's level of expertise. *Computational Linguistics*, 14:64–78, 1988.

[Reithinger, 1991] N. Reithinger. *Eine parallele Architektur zur inkrementellen Generierung multimodaler Dialogbeiträge.* PhD thesis, Universität des Saarlandes, Saarbrücken, 1991.

[Sibun, 1990] P. Sibun. The local organization of text. In K. R. McKeown, J. D. Moore, and S. Nirenburg, eds., *Proc. of 5th International Natural Language Generation Workshop*, pages 120–127, 1990.