Paraphrasing and Aggregating Argumentative Text Using Text Structure

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

Abstract

We argue in this paper that sophisticated microplanning techniques are required even for mathematical proofs, in contrast to the belief that mathematical texts are only schematic and mechanical. We demonstrate why paraphrasing and aggregation significantly enhance the flexibility and the coherence of the text produced. To this end, we adopted the Text Structure of Meteer as our basic representation. The type checking mechanism of Text Structure allows us to achieve paraphrasing by building comparable combinations of linguistic resources. Specified in terms of concepts in an uniform ontological structure called the Upper Model, our semantic aggregation rules are more compact than similar rules reported in the literature.

1 Introduction

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 [12] for an extensive discussion). An increasing interest in more sophisticated microplanning techniques can be clearly observed [12, 14], however. In this paper, we first motivate the needs for paraphrasing and aggregation for the generation of argumentative texts, in particular of mathematical proofs, and then describe how our microplanning operations can be formulated in terms of Meteer's *Text Structure*.

The work reported here is part of a fully implemented system called *PROVERB*, which produces natural language proofs from proofs found by automated reasoning systems [7]. First experiments with *PROVERB* resulted in very mechanical texts due to the lack of microplanning techniques. According to our analysis, there are at least two linguistic phenomena that call for appropriate microplanning techniques.

First, naturally occurring proofs contain paraphrases with respect to both 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 exactly mirroring the information structure of the proof and the formulae. This means that every step of derivation 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 flexible combinations of linguistic resources. It is worth pointing out that these techniques are required although the input information chunks are of clause size. Another requirement is that this intermediate representation is easy to control, since a mathematical text must conform to the syntactic rules of its sublanguage. In the next section, we first give a brief overview of *PROVERB*. Then we describe the architecture of our microplanner, and illustrate how Meteer's Text Structure can be adopted as our central representation. In Sec. 5 and 6 we describe the handling of paraphrases and aggregation rules, two of the major tasks of our microplanner.

2 The Macroplanner of *PROVERB*

The macroplanner of *PROVERB* combines *hierarchical planning* [13] with *local organization* [15] in a uniform planning framework [6]. The hierarchical planning is realized by so-called top-down presentation operators that split the task of presenting a particular proof into subtasks of presenting subproofs. While the overall planning mechanism is similar to the RST-based planning approach, the plan operators resemble the schemata in *schema-based* planning. The output of the macroplanner is an ordered sequence of *proof communicative acts* (PCAs).

PCAs are the primitive actions planned during macroplanning to achieve communicative goals. Like speech acts, PCAs can be defined in terms of the communicative goals they fulfill as well as in terms of their possible verbalizations. Based on an analysis of proofs in mathematical textbooks, there are mainly two types of goals:

Conveying derivation step: In terms of rhetorical relations, PCAs in this category represent a variation of the rhetorical relation *derive* [8]. Below we examine the simple PCA called `Derive` as an example.

```
(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."

Updating the global attentional structure: These PCAs either convey a partial plan for the forthcoming discourse or signal the end of a subproof. PCAs of this sort are also called *meta-comments* [16].

The PCA

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

produces the verbalization:

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

3 Text Structure in *PROVERB*

3.1 Introduction and General Structure

Text Structure is first proposed by Meteer [11, 12] in order to bridge the generation gap between the representation in the application program and the linguistic resources provided by the language. By abstracting over concrete linguistic resources, Text Structure should supply the planner with basic vocabularies, with which it chooses linguistic resources. Meteer's text structure is organized as a tree, in which each node represents a constituent of the text. In this form it contains three types of linguistic information: *constituency, structural relations among constituents*, and in particular, the *semantic categories the constituents express*.

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 builds 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. Panaget [14] argues, however, that Meteer's semantic categories mix the ideational and the textual dimension as argued in the systemic linguistic theory [5]. Here is one of his examples:

"The ship sank" is an ideational *event*, and it is textually presented from an EVENT-PERSPECTIVE. "The sinking of the ship" is still an ideational *event*, but now presented from an OBJECT-PERSPECTIVE.

On account of this, Panaget split the type restrictions into two orthogonal dimensions: the ideational dimension in terms of the *Upper Model* [1], and the *hierarchy of textual semantic categories* based on an analysis of French and of English. In our work, we basically follow the approach of Panaget.

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.

3.2 Type Restrictions

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. Figure 1 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.

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[1] is realized as an adverb, clause-modifier-rankingII as a prepositional phrase, and clause-modifier-embedded as an adverbial clause. Fig. 2 shows a fragment of the hierarchy of textual semantic categories.

3.3 Mapping APOs to UMOs

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. It is worth noting that there is a practical advantage of this two-staged process. 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 resource trees of an Upper Model concept are assembled in its *realization class.*

[1] Concepts of the hierarchy of textual semantic categories are noted in sans-serif text.

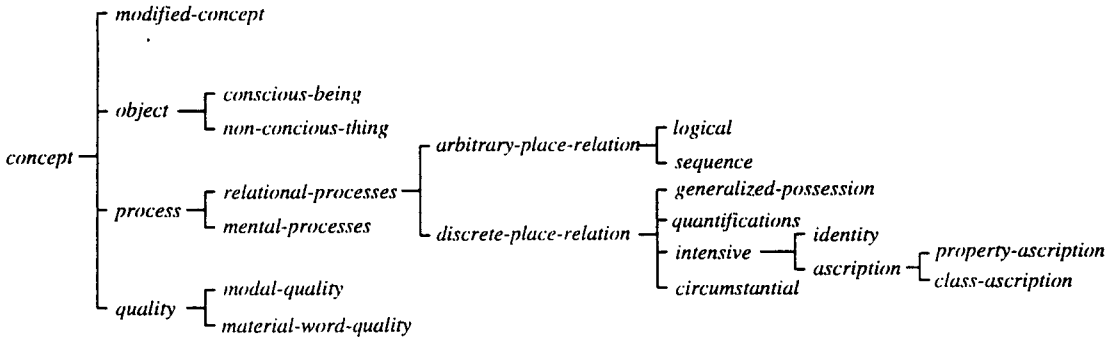


Figure 1: A Fragment of Upper Model in *PROVERB*

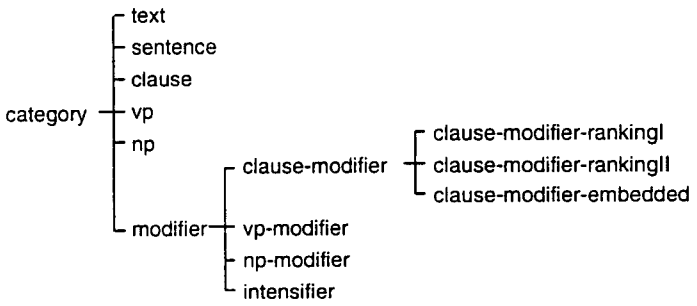


Figure 2: A Fragment of the Hierarchy of Textual Semantic Categories in *PROVERB*

4 Architecture and Control

The main tasks of our microplanner include aggregation to remove redundancies, insertion of cue words to increase coherence, and reference choices, as well as lexical choices. Apart from that, the microplanner also handles sentence scoping and layout. An overview of the microplanner's architecture is provided in Figure 3.

Our microplanner takes as input an ordered sequence of PCAs, structured in an attentional hierarchy. The first module, the *derivation reference choice component* (DRCC), suggests which parts of a PCA are to be verbalized. This is done based on the hierarchical discourse structure as well as on the textual distance. PCAs annotated with these decisions annotated are called *preverbal messages* (PMs).

Starting from a list of PMs as the initial Text Structure, the microplanner progressively maps application program concepts in PMs into text structure objects of some textual semantic type by referring to Upper Model objects as an intermediate level. The Text Structure evolves by the expansion of leaves top-down and left to right. This process is controlled by the main module of our microplanner, the *Text Structure Generator* (TSG), which carries out the following algorithm:

- When the current node is an APO with more than one son, apply ordering and aggregation, in order to produce more concise and more coherent text. The application of an aggregating rule before the expansion of a leaf node may trigger the insertion of cue words.
- An APO is mapped into an UMO, which is in turn expanded into a Text Structure by choosing an appropriate resource tree.

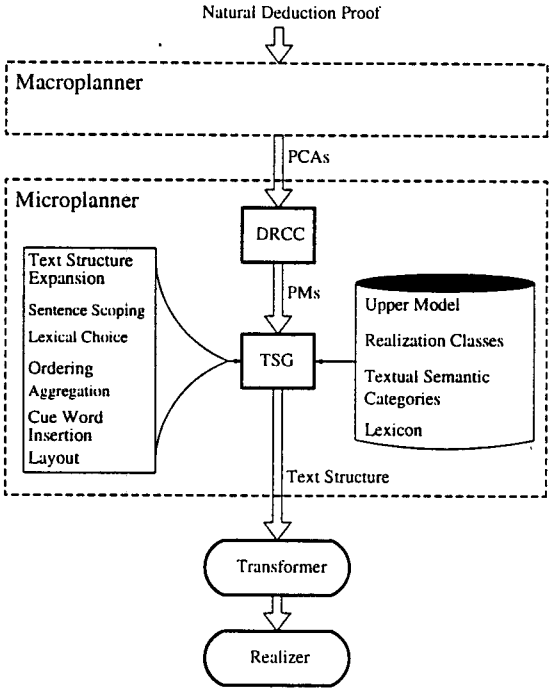


Figure 3: Architecture of the Microplanner

- A fully expanded Text Structure will be traversed again:
  - — to choose the appropriate lexical items.
  - — to make sentence scoping decisions by singling out one candidate textual semantic category for each constituent. This in turn may trigger the execution of a cue word rule. For instance, the choice of the category *sentence* for a constituent may lead to the insertion of the cue word "furthermore" in the next sentence.
  - — to determine the layout parameters, which will be realized later as LaTeX-commands in the final output text.

A Text Structure constructed in this way is the output of our microplanner, and will be transformed into the input formalism of TAG-GEN [10], our linguistic realizer.

In the next two sections, we concentrate on two major tasks of the Text Structure generator: to choose compatible paraphrases of application program concepts, and to improve the textual structure by applying aggregation rules.

5 Paraphrasing in *PROVERB*

With the help of a concrete example we illustrate in this section how the Text Structure generator chooses among paraphrases and avoids building inexpressible text structures via type checking.

Example We examine a simple logic formula *derive*(`para(C1,C2)`,B). Note that B stands for a conclusion which will not be examined here. We will also not follow the procedure in detail.

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 leading to different patterns of verbalization. We only list two variations below:

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

The resource tree of the first alternative is given in Fig. 4.

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

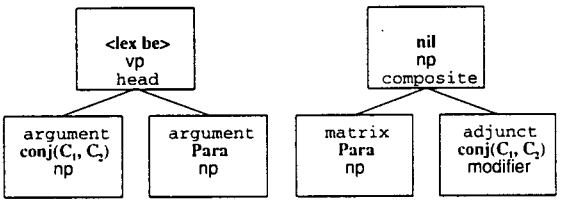
Textually, the property-ascription version can be realized in two forms, represented by the two resource trees in Fig. 5.

Type checking during the construction of the Text Structure must ensure, that the realization be compatible along both the ideational and the textual dimension. In this example, the combination of the tree in Fig. 4 and the first tree in Fig. 5 is compatible and will lead to the verbalization:

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

```
(realization-class (derive :reason R :conclusion C)
  (resource-tree (composite-tree :content nil
                                 :tsc (sentence clause)
                                 :matrix (leaf :content C
                                               :tsc (clause))
                                 :adjunct (composite-tree :content since
                                                          :tsc (clause)
                                                          :matrix (leaf :content R
                                                                        :tsc (clause)))))
  (further resource trees ...))
```

Figure 4: The Realization Class for *derive*



As a verb phrase As a nominal phrase

Figure 5: Textual Variations in form of Resource Trees

The second tree in Fig. 5, 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 concentrat on the mechanism and are therefore still experimenting with heuristics which 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, continuity of the latter helps the user to identify technical concepts.

6 Semantic Aggregation Rules

Although the handling of 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 forced to keep these structures. To achieve the second verbalization of equation (1) in the introduction, however, we have to combine $Set(F)$ and $Subset(F, G)$ to form an embedded structure $Subset(Set(F), G)$. Clearly, although still in the same format, this is no more an Upper Model object, since $Set(F)$ is an Upper Model *process*, not an *object*. Actually, this documents a textual decision that no matter how *Subset* and *Set* should be instantiated, the argument F in $Subset(F, G)$ will be replaced by $Set(F)$. This textual operation eliminates one of the duplicates of F. This section is devoted to various textual reorganisations which eliminate such redundancies. Following the tradition, we call them *aggregation* rules.

As it will become clear when handling concrete aggregation rules, such rules may narrow the realization choices of APOs by imposing additional type restrictions. Furthermore, some realization choices block desirable textual reorganisation. On account of this we carry out aggregations before concrete resources for the APOs like *object* and *class-ascription* are chosen.

APOs, before they are mapped to UMOs, can be viewed as variables for UMOs (for convenience, we continue to refer to them as APOs). In this sense, our rules work with such variables at the *semantic level* of the Upper Model, and therefore differ from those more syntactic rules reported in the literature. For a comparison see Sec. 6.4.

So far, we have investigated three types of aggregation which will be addressed in the next two subsections. A categorization of the aggregation rules is given in Fig. 6.

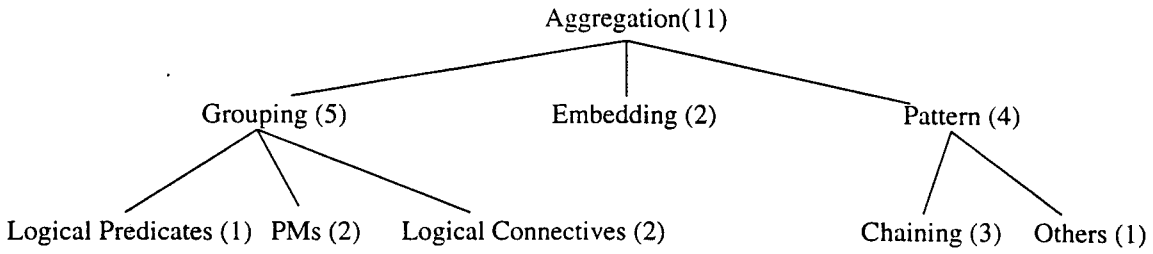


Figure 6: Aggregation Rules in *PROVERB*

6.1 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

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

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 $\oplus$. In the first rule below, $+$ and $\oplus$ 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 [3]. 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.

Now let us examine another semantic grouping rule, where $+$ and $\oplus$ are no longer identical.

Rule A.2 (Implication with identical conclusion)

$$\frac{(P_1 \rightarrow C) \wedge (P_2 \rightarrow C)}{(P_1 \vee P_2) \rightarrow C}$$

Here $+$, $\oplus$, and P are instantiated to $\wedge$, $\vee$, and $\rightarrow$, respectively. By instantiating $+$, $\oplus$ and P in pattern A to different logical connectives and derivation relations, we have alltogether 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 Upper Model concerned. In the case of rule A.2 for instance, $(P_1 \vee P_2) \rightarrow C$ is a logical consequence of $(P_1 \rightarrow C) \wedge (P_2 \rightarrow C)$.

6.2 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)

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

where

- $concepts(f, T) \cap concepts(P) \neq \emptyset$

- f is the innermost application program concept governing T in $Q[T]$,
- *concepts*(f, T) denotes the Upper Model concepts the argument T of f may take,
- *concepts*(P) denotes the Upper Model concept P may result in.

We require also that $P[T]$ is realized as an object T with modifiers. It is this intuitive explanation which guarantees the correctness of this rule with respect to meaning.

The following example illustrates this rule, in particular, how the decision made here narrows the choices of linguistic resources for both P and T as an argument of Q. We begin with the two APOs in a conjunction below, containing a common APO F.

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

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

Since F is directly governed by *Subset*, f and Q in our rule above coincide here. *concepts*$(Subset, F)$ = {*object*}, while *concepts*(Set) = {*class-ascription*, *object*}. Therefore, their intersection is {*object*}. This not only guarantees the expressibility of the new APO, but also restricts the choice of linguistic resources for *Set*, now restricted to *object*. The result as well as its verbalization is given below:

Subset(*Set*$(F), G)$

"The set F is a subset of G."

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

6.3 Pattern-based Optimization rules

Rules in the third category involve more complex changes of the textual structure in a way which is neither a grouping nor an embedding. They could be understood as some domain-specific communicative conventions, and must be explored in every domain of application. In *PROVERB*, currently four such rules are integrated. Three of them build a sequence of some transitive relations into a chain.

Rule C.1 below addresses the problem that every step of derivation is mapped to a separate sentence in the default verbalization. It reflects the familiar phenomenon that when several derivation steps form a chain, they are verbalized in a more connected way. To accommodate the phenomenon of a chain, we have also added a slot called *next* in the domain model concept *derive-chain*. Now suppose that we have two consecutive derivations with R_1, M_1, C_1 and R_2, M_2, C_2 as its premises (called reasons), the rule of inference (called method), and the conclusion. They form part of a chain if the conclusion C_1 is used as a premise in the second step, namely $C_1 \in R_2$. In this case, the following rule combines them into a chain by putting the second derivation into the *next* slot of the chain. At the same time, C_1 is removed from R_2 since it is redundant.

Rule C.1 Derivation Chain[2]

$$\frac{\textit{derive}(R_1, M_1, C_1), \textit{derive}(R_2, M_2, C_2)}{\textit{derive-chain}(R_1, M_1, C_1, \textit{derive}(R_2 \backslash C_1, M_2, C_2,))}$$

The following example illustrates how this rule works. We will only give the verbalization and omit the Text Structure. Given a sequence of two derivation steps which can be verbalized as:

> "$\sigma \subseteq \sigma^*$, by the definition of transitive closure." and
> "Since $(x, y) \in \sigma$ and $\sigma \subseteq \sigma^*$, $(x, y) \in \sigma^*$ by the definition of subset."

Rule C.1 will produce a chain which will be verbalized as

> "$\sigma \subseteq \sigma^*$ by the definition of transitive closure, thus establishing $(x, y) \in \sigma^*$ by the definition of subset, since $(x, y) \in \sigma$."

Note that the rule above is only a simplification of a recursive definition, since chaining is not restricted to two derivation steps.

[2]This is a simplified version of the original rule defined recursively in [4]

Readers are referred to [4]. Although this rule inserts the second *derive* into another Text Structure, the resulting structure is now a chain, no longer a plain *derive*. Therefore it distinguishes clearly from the rules in Section 6.2.

There are two more chaining rules for the logical connectors *implication* and *equivalence*. A further rule removes redundancies in some case analyses (see [4]).

6.4 Discussion

While many systems have some aggregation rules implemented [9, 2], there are comparatively few detailed discussions in the literature. The most structured categorization we found is the work of Dalianis and Hovy [3], where they define aggregation as a way of avoiding redundancy. Some of their rules, nevertheless, make decisions which we would call reference choice. Since this is treated in another module, we define our aggregation at the semantic level. The following are several significant features of our aggregation rules.

The first difference is that our aggregation rules are defined in terms of manipulations of the Upper Model. They remove redundancies by combining the linguistic resources of two adjacent APOs, which contain redundant content. They cover the more syntactic rules reported in the literature at a more abstract level.

Second, Text Structure provides us stronger means to specify textual operations. While rules reported in the literature typically aggregate clauses, our rules operate both above and beneath the level of clause constituents.

Third, while most investigations have concentrated on general purpose microplanning operations, we came to the conclusion that microplanning needs domain-specific rules and patterns as well.

7 A Running Example

The following example illustrates the mechanism of aggregation and its effect on resulting text. We start with the following sequence of PMs:

$$\begin{aligned}
&\mathit{assume}(\mathit{Set}(F))\\
&\mathit{assume}(\mathit{Set}(G))\\
&\mathit{assume}(\mathit{Subset}(F,G))\\
&\mathit{assume}(\mathit{element}(a,F))\\
&\mathit{assume}(\mathit{element}(b,F))\\
&\mathit{derive}((\mathit{element}(a,F) \wedge \mathit{Subset}(F,G)),\\
&\qquad \epsilon, \mathit{element}(a,G))\\
&\mathit{derive}((\mathit{element}(b,F) \wedge \mathit{Subset}(F,G)),\\
&\qquad \epsilon, \mathit{element}(b,G))
\end{aligned}$$

Without aggregation, the system produces:

"Let F be a set. Let G be a set. Let $F \subset G$. Let $a \in F$. Let $b \in F$. Since $a \in F$ and $F \subset G$, $a \in G$. Since $b \in F$ and $F \subset G$, $b \in G$."

Aggregation of the *assume*-PMs results in:

$$\begin{aligned}
&\mathit{assume}(\mathit{Set}(F) \wedge \mathit{Set}(G) \wedge \mathit{Subset}(F,G)\\
&\qquad \wedge \mathit{element}(a,F) \wedge \mathit{element}(b,F))
\end{aligned}$$

whereas the application of the grouping rule for independent *derive*-PMs provides:

$$\begin{aligned}
&\mathit{derive}((\mathit{element}(a,F) \wedge \mathit{element}(b,F)\\
&\qquad \wedge \mathit{Subset}(F,G)), \epsilon,\\
&\qquad (\mathit{element}(a,G) \wedge \mathit{element}(b,G)))
\end{aligned}$$

After that, the predicate grouping rule A.1 is applied to the arguments of *assume*, which are grouped to:

$$\begin{aligned}
&(\mathit{Set}(F \wedge G) \wedge \mathit{Subset}(F,G)\\
&\quad \wedge \mathit{element}(a \wedge B, F \wedge F)))
\end{aligned}$$

Note that $F \wedge F$ is later reduced to F. Predicate grouping applies to the arguments of *derive* in a similar way. Finally, the system produces the following output:

"Let F and G be sets, $F \subset G$, and $a, b \in F$. Then $a, b \in G$."

8 Conclusion

We argued in this paper that sophisticated microplanning techniques are required even for mathematical proofs, in contrast to the belief that mathematical texts are only schematic and mechanical. We demonstrated why paraphrasing and aggregation will significantly enhance the flexibility and the coherence of text produced. In order to

carry out appropriate textual rearrangement we need a representation formalism which allows flexible but principled manipulation of linguistic resources. To this end, we basically adopted the Text Structure of Meteer, but split her semantic categories into two dimensions following Panaget. The type checking mechanism of Text Structure allows us to achieve paraphrasing by building comparable combinations of linguistic resources. Specified in terms of Upper Model concepts, our semantic aggregation rules are more abstract than similar rules reported in the literature.

One important feature of our work is the integration of microplanning knowledge specific to our domain of application. This body of knowledge must be refined to further improve the quality of the text produced. More experience is also required to formulate strategies to choose among alternatives.

References

[1] John Bateman, Bob Kasper, Johanna Moore, and Richard Whitney. The penman upper model. Technical Report ISI research report, USC/Information Science institute, 1990.

[2] Robert Dale. *Generating Referring Expressions.* ACL-MIT PressSeries in Natural Language Processing. MIT Press, 1992.

[3] Hercules Dalianis and Eduard Hovy. Aggregation in natural language generation. In *Proc. 4th European Workshop on Natural Language Generation*, 1993.

[4] Armin Fiedler. Mikroplanungstechniken zur Präsentation mathematischer Beweise. Master's thesis, Fachbereich Informatik, Universität des Saarlandes, 1996.

[5] M.A.K. Halliday. *Introduction to functional grammar.* Edward Arnold, 1985.

[6] Xiaorong Huang. Planning argumentative texts. In *Proc. of 15th International Conference on Computational Linguistics*, 1994.

[7] Xiaorong Huang. *PROVERB*: A system explaining machine-found proofs. In *Proc. of 16th Annual Conference of the Cognitive Science Society*, 1994.

[8] Xiaorong Huang. *Human Oriented Proof Presentation: A Reconstructive Approach.* Infix, 1996.

[9] Gerard Kempen. Conjunction reduction and gapping in clause-level coordination: an inheritance-based approach. *Computational Intelligence*, 7(4):357–360, 1991.

[10] Anne Kilger and Wolfgang Finkler. Incremental generation for real–time applications. Research Report RR-95-11, DFKI, Saarbrücken, 1995.

[11] Marie W. Meteer. Bridging the generation gap between text planning linguistic realization. *Computational Intelligence*, 7(4), 1991.

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

[13] Johanna Doris Moore and Cécile L. Paris. Planning text for advisory dialogues. In *Proc. 27th Annual Meeting of the Association for Computational Linguistics*, 1989.

[14] Franck 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*, 1994.

[15] Penelope Sibun. The local organization of text. In *Proc. of the fifth international natural language generation workshop*, 1990.

[16] Ingrid Zukerman. Using meta-comments to generate fluent text in a technical domain. *Computational Intelligence*, 7:276–295, 1991.