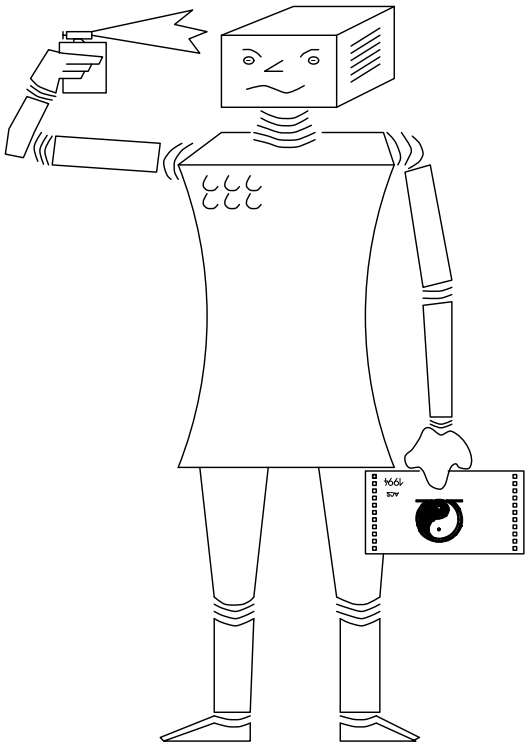


Aggregation in the Generation of Argumentative Texts

Armin Fiedler and Xiaorong Huang

Published as: Proc. of 5th European Workshop on Natural Language Generation

Aggregation in the Generation of Argumentative Texts

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

1 Introduction

One of the most important techniques for sentence planning is *aggregation*. Without loss of information, aggregation reorganizes and merges information items at various levels to eliminate redundancy and to produce more streamlined text. Although not always studied in great depth, corresponding techniques have been implemented in assorted systems. Based on an empirical study, Dalianis and Hovy proposed a classification of aggregation rules [2].

In this paper, we explore the aggregation technique within the system *PROVERB*, which verbalizes machine-found proofs. Most of our rules can be seen as adaptations and extensions of the grouping rules identified in [2] for argumentative texts. A further class of rules is introduced, called *chaining*. Altogether, eleven rules stated at the discourse and semantic level are integrated. We will show, that the incorporation of aggregation significantly improves the coherence of text produced.

2 The Architecture of *PROVERB*

The macroplanner of *PROVERB* combines *hierarchical planning* with *local organization* in a uniform planning framework. The output is an ordered sequence of *proof communicative acts* (PCAs) [4]. The simplest PCA conveying a step of derivation is called `Derive`. Below is an instance:

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

Depending on the reference choices, it may be verbalized as:

> "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."

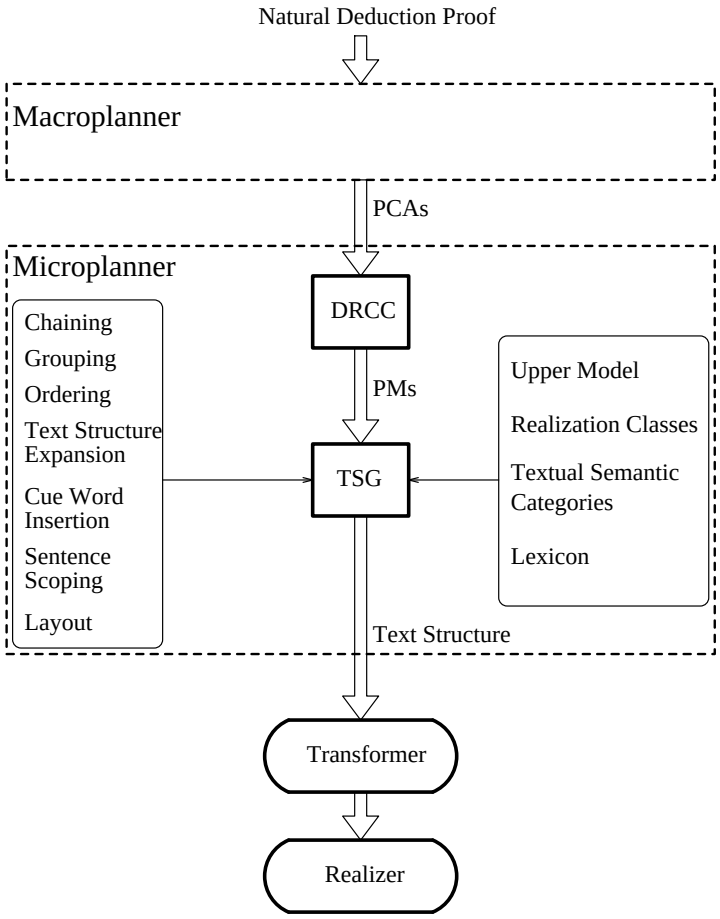
Our microplanner is based on an intermediate representation called *text structure*, proposed by Meteer [5]. This text structure reflects linguistic constraints, while abstracting away from syntactic detail. Following Panaget, we split Meteer's hierarchy of semantic categories into two orthogonal hierarchies: an ideational semantic hierarchy

based on the *upper model* defined in [1], and a textual semantic hierarchy, which Panaget calls *hierarchy of textual semantic categories* [6]. A domain concept is inserted to the upper model for every concept used in the PMs, primarily including the rhetorical relations, predicates, and function symbols in predicate logic.

The content of our text structure nodes is called a *text structure object*, which can be one of the following: an upper model object, an application object (such as PMs and formulae), as well as a compound upper model object with application objects as leaves.

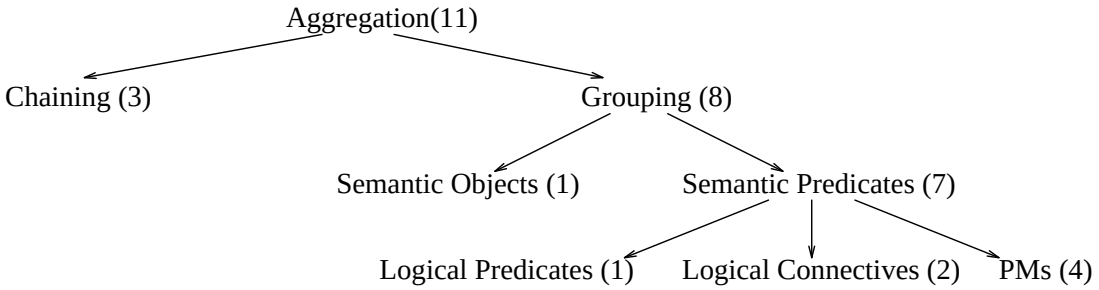
Our microplanner takes as input an ordered sequence of PCAs. The first module is the *derivation reference choice component* (DRCC). It suggests which parts of a PCA are to be verbalized. PCAs with these decisions annotated are called *preverbal messages* (PMs).

Starting from a single-node text structure containing a list of preverbal messages, the main module of our microplanner, namely the *text structure generator* (TSG), progressively maps concepts in preverbal messages into appropriate linguistic realizations via upper model objects as an intermediate level. The text structure evolves by expanding leaves top-down and from left to right. While doing so, TSG draws on various rules, including aggregation to remove redundancies, insertion of cue words to increase coherence, lexical choices, sentence scoping and layout. A text structure constructed in this way is the output of our microplanner, and will be transformed into the input formalism of TAG-GEN, our linguistic realizer. An overview of the microplanner's architecture is provided in the following figure.



3 Aggregation

The figure below shows a first classification of our aggregation rules. With eleven rules in total, and defined exclusively in terms of text structure objects, this represents a refinement of the classification in [2]. As a powerful extension for mathematical text, three *chaining* rules are identified to construct derivation chains. Part of the rules in [2] are handled by other components in our system: the dynamic rule by our reference choices module, and the economy and the casting rules by our module performing realization class choices. Note that the grouping rules work in conjunction with ordering strategies. In this paper, we concentrate on some of the most representative rules. For a complete listing, readers are referred to [3].



If the conclusion of a *derive*-PM is a reason of the next, these *derive*-PMs can be nested into a *derivation chain*, which is defined as follows:

(i) Each *derive*-PM is a derivation chain of length 1.

(ii) $derive\langle\Phi_1, derive\langle\Phi_2, \cdots derive\langle\Phi_{k-1}, derive\langle\Phi_k\rangle\rangle \cdots\rangle\rangle$ is a derivation chain of length k, where Φ_i stands for the sequence R_i, M_i, C_i of the reasons, method and conclusion. The conclusion of $derive\langle\Phi_{i-1}, \ldots\rangle$ is logically a reason in $derive\langle\Phi_i, \ldots\rangle$.

The nesting is carried out by the following rule:

Rule 1 *(chaining)*
Let R_1 be the reasons, M_1 the method and C_1 the conclusion of the last *derive*-PM in a derivation chain (possibly of length 1). Furthermore, let R_2 be the reasons, M_2 the method and C_2 the conclusion of another *derive*-PM. If $C_1 \in R_2$, then extend the derivation chain by one *derive*-PM:

$$\frac{derive\langle\cdots derive\langle R_1, M_1, C_1\rangle\cdots\rangle, derive\langle R_2, M_2, C_2\rangle}{derive\langle\cdots derive\langle R_1, M_1, C_1, derive\langle R_2[\epsilon/C_1], M_2, C_2\rangle\rangle\cdots\rangle}$$

Intuitively, since C_1 is just verbalized as the last conclusion, it will not be verbalized explicitly as a reason. $R_2[\epsilon/C_1]$ indicates that C_1 is erased from R_2.

Example:
$derive\langle\epsilon, DefTrans, \underline{Subset(\sigma, transclosure(\sigma))}\rangle$
$derive\langle(\sigma(x,y), \underline{Subset(\sigma, transclosure(\sigma))}, DefSubset, transclosure(\sigma)(x,y)\rangle$
"We can derive $\underline{\sigma \subset \sigma^*}$ by definition of the transitive closure. Since $(x,y) \in \sigma$ and $\underline{\sigma \subset \sigma^*}$, $(x,y) \in \sigma^*$ by the definition of subset."

$\Downarrow$

$derive\langle\epsilon, DefTrans, \underline{Subset(\sigma, transclosure(\sigma))},$
$\quad derive\langle(\sigma(x,y)), DefSubset, transclosure(\sigma)(x,y)\rangle\rangle$
"We can derive $\underline{\sigma \subset \sigma^*}$ by the definition of the transitive closure, which gives us $(x,y) \in \sigma^*$ by the definition of subset since $(x,y) \in \sigma$."

The next rule we introduce removes redundant objects.

Rule 2 *(object grouping)*
Let P and Q be upper model concepts subordinate to the concept *object*. Furthermore, let $S_1, \ldots, S_m$ and $T_1, \ldots, T_n$ be application objects. If $S_k \in \{T_1, \ldots, T_n\}$ for a $1 \leq k \leq m$:

$$\frac{P\langle S_1, \ldots, S_{k-1}, S_k, S_{k+1}, \ldots, S_m\rangle, Q\langle T_1, \ldots, T_n\rangle}{P\langle S_1, \ldots, S_{k-1}, Q\langle T_1, \ldots, T_n\rangle, S_{k+1}, \ldots, S_m\rangle}$$

Note that the application of this rule restricts the choice of realization classes.
Example:
$Subset\langle F, G\rangle$, $Set\langle F\rangle$
"$\underline{F}$ is a subset of G. $\underline{F}$ is a set."
$\Downarrow$
$Subset\langle Set\langle F\rangle, G\rangle$
"The set $\underline{F}$ is a subset of G."

The following rule merges assertions with the same predicate.

Rule 3 *(Grouping of Logical Predicates)*
Let P be a predicate of predicate logic.

$$\frac{P\langle S_{1,1}, \ldots, S_{1,n}\rangle, \ldots, P\langle S_{k,1}, \ldots, S_{k,n}\rangle}{P\langle (S_{1,1}, \ldots, S_{k,1}) \ldots, (S_{1,n}, \ldots, S_{k,n})\rangle}$$

Example:
$Set\langle F\rangle$, $Set\langle G\rangle$
"F is a set. G is a set."
$\Downarrow$
$Set\langle (F, G)\rangle$
"F and G are sets."

The next rule represents a fairly complex pattern, which merges two similar branches of a case analysis.

Rule 4 *(similar cases)*
Let $M_i = M_i'$ for all $1 \leq i \leq n$, $C_n = C_n'$ and P and P' the case assumptions. Furthermore, let $P \in R_k$ and $P' \in R_k'$ for a $1 \leq k \leq n$.

$$\frac{\begin{array}{r} Begin\text{-}Cases, Case1\langle P\rangle\, , derive\langle R_1, M_1, C_1, \cdots derive\langle R_n, M_n, C_n\rangle \cdots\rangle, \\ Case2\langle P'\rangle, derive\langle R_1', M_1', C_1', \cdots derive\langle R_n', M_n', C_n'\rangle \cdots\rangle, End\text{-}Cases \end{array}}{derive\langle R_1'', M_1'', C_1'', \cdots derive\langle R_n'', M_n'', C_n''\rangle \cdots\rangle}$$

where

$$R_i'' = \begin{cases} R_i \circ R_i' & \text{if } 1 \leq i < k \\ (R_i[\epsilon/P]) \circ (R_i'[\epsilon/P']) \circ (\vee\langle (P, P')\rangle) & \text{if } i = k \\ \vee\langle (\wedge\langle R_i\rangle, \wedge\langle R_i'\rangle)\rangle & \text{if } k < i \leq n \end{cases}$$

$$M_i'' = M_i \quad \text{for all } 1 \leq i \leq n$$

$$C_i'' = \begin{cases} \wedge\langle (C_i, C_i')\rangle & \text{if } 1 \leq i < k \\ \vee\langle (C_i, C_i')\rangle & \text{if } k \leq i < n \\ C_n & \text{if } i = n \end{cases}$$

Here $\circ$ stands for concatenation. Since a list of reasons in a *derive*-PM is logically a conjunction, this must be made explicit if used as an argument in a more complex formula, as in the third case of R''.

Example:
Begin-Cases,
$Case1\langle F_1(a)\rangle$, $derive\langle(F_1(a), Subset(F_1, G)), DefSubset, G(a)\rangle$,
$Case2\langle F_2(a)\rangle$, $derive\langle(F_2(a), Subset(F_2, G)), DefSubset, G(a)\rangle$,
End-Cases
"Let us consider the first case, where $a \in F_1$. Since $a \in F_1$, $a \in G$ by the definition of subset. Now, let us consider the second case, where $a \in F_2$. Since $a \in F_2$, $a \in G$ by the definition of subset."

$\Downarrow$

$derive\langle\vee\langle(\wedge\langle(F_1(a), Subset(F_1, G))\rangle, \wedge\langle(F_2(a), Subset(F_2, G))\rangle)\rangle$,
$DefSubset, G(a)\rangle$
"Since either $a \in F_1 \subset G$ or $a \in F_2 \subset G$, $a \in G$ by the definition of the subset."

4 A Working Example

We prove $(\rho \cup \sigma)^* \subseteq (\rho^* \cup \sigma^*)^*$.
(1) Let $(x_1, y_1) \in \rho \cup \sigma$. **(2)** Then $(x_1, y_1) \in \rho$ or $(x_1, y_1) \in \sigma$.
(3) Let us consider the first case by assuming that $(x_1, y_1) \in \rho$. **(4)** By the definition of the transitive closure $\rho \subseteq \rho^*$. **(5)** Therefore $(x_1, y_1) \in \rho^*$. **(6)** Hence $(x_1, y_1) \in \rho^*$ or $(x_1, y_1) \in \sigma^*$. **(7)** Now let us consider the second case by assuming that $(x_1, y_1) \in \sigma$. **(8)** By the definition of the transitive closure $\sigma \subseteq \sigma^*$. **(9)** Therefore $(x_1, y_1) \in \sigma^*$. **(10)** Hence $(x_1, y_1) \in \rho^*$ or $(x_1, y_1) \in \sigma^*$. **(11)** Since $(x_1, y_1) \in \rho^*$ or $(x_1, y_1) \in \sigma^*$ $(x_1, y_1) \in \rho^* \cup \sigma^*$. **(12)** By the definition of the transitive closure $(\rho^* \cup \sigma^*) \subseteq (\rho^* \cup \sigma^*)^*$. **(13)** Therefore $(x_1, y_1) \in (\rho^* \cup \sigma^*)^*$. **(14)** Hence $(\rho \cup \sigma) \subseteq (\rho^* \cup \sigma^*)^*$. **(15)** By the transitivity of the transitive closure $(\rho^* \cup \sigma^*)^*$ is transitive. **(16)** Therefore $(\rho \cup \sigma)^* \subseteq (\rho^* \cup \sigma^*)^*$ by the minimality of the transitive closure.

Aggregation $\Downarrow$

We prove $(\rho \cup \sigma)^* \subseteq (\rho^* \cup \sigma^*)^*$.
(1,2) Let $(x_1, y_1) \in \rho \cup \sigma$, which implies that $(x_1, y_1) \in \rho$ or $(x_1, y_1) \in \sigma$. **(3-11)** $\rho \subseteq \rho^*$ and $\sigma \subseteq \sigma^*$ by the definition of the transitive closure, thus establishing $(x_1, y_1) \in \rho^*$ or $(x_1, y_1) \in \sigma^*$, which gives us $(x_1, y_1) \in \rho^* \cup \sigma^*$. **(12,13)** Since $(\rho^* \cup \sigma^*) \subseteq (\rho^* \cup \sigma^*)^*$ by the minimality of the transitive closure, $(x_1, y_1) \in (\rho^* \cup \sigma^*)^*$. **(14-16)** Since $(\rho^* \cup \sigma^*)^*$ is transitive by the transitivity of the transitive closure and $(\rho \cup \sigma) \subseteq (\rho^* \cup \sigma^*)^*$, $(\rho \cup \sigma)^* \subseteq (\rho^* \cup \sigma^*)^*$ by the minimality of the transitive closure.

References

1. J. Bateman, B. Kasper, J. Moore, and R. Whitney. The penman upper model. Technical Report ISI research report, USC/ISI, 1990.

2. H. Dalianis and E. Hovy. Aggregation in natural language generation. *Proc. of 4th European Workshop on Natural Language Generation*, 1993.

3. A. Fiedler and X. Huang. Aggregation for mathematical proofs. Seki-Report, Fachbereich Informatik, Universität des Saarlandes, 1995. forthcoming.

4. X. Huang. Planning argumentative texts. In *Proc. of 15th COLING*, Kyoto, 1994.

5. M. Meteer. *Expressibility and the Problem of Efficient Text Planning.* Pinter Publishes, 1992.

6. 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*, Kennebunkport, USA, 1994.