De-Constraining Text Generation

Stephen Beale, Sergei Nirenburg, Evelyne Viegas† and Leo Wanner§

†Computing Research Laboratory
Box 30001, Dept. 3CRL
New Mexico Sate University
Las Cruces, NM 88003-0001 USA
sb,sergei,viegas@crl.nmsu.edu

§Computer Science Department, IS
University of Stuttgart
Breitwiesenstr. 20-22
D-70565 Stuttgart, Germany
wannerlo@informatik.uni-stuttgart.de

Abstract

We argue that the current, predominantly task-oriented, approaches to modularizing text generation, while plausible and useful conceptually, set up spurious conceptual and operational constraints. We propose a data-driven approach to modularization and illustrate how it eliminates the previously ubiquitous constraints on combination of evidence across modules and on control. We also briefly overview the constraint-based control architecture that enables such an approach and facilitates near linear-time processing with realistic texts.

1 Introduction

This paper addresses the area of text generation known as *microplanning* [Levelt1989, Panaget1994, Huang and Fiedler1996], or *sentence planning* [Rambow and Korelsky1992]; [Wanner and Hovy1996]. Microplanning involves low-level discourse structuring and marking, sentence boundary planning, clause internal structuring and all of the varied subtasks involved in lexical choice. These complex tasks are often modularized and treated separately. The general argument is that since sentence planning tasks are not single-step operations, since they do not have to be performed in strict sequence, and since the planner's operation is non-deterministic, each sentence planning task should be implemented by a separate module or by several modules (see, e.g., [Wanner and Hovy1996]). Such an argument is natural if generation is viewed as a set of coarse-grained tasks. Indeed, with the exception of a few researchers ([Elhadad et al.1997] and the incrementalists listed below), the task-oriented view is standard in the generation community. Unfortunately, task-oriented generation sets up barriers among the components of the generation process, primarily because, in a realistic scenario, the tasks are intertwined to a high degree. Overcoming these barriers has become a central topic in generation research (see below). In our approach the basis of modularization is sought in the nature of the input data to the generation process, in our case, a text meaning representation, formulated largely in terms of an ontology. This data-oriented approach is similar to that taken by many incremental generators [De Smedt1990, Reithinger1992], although these tend to concentrate on syntactic processing. But see [Kilger1997], who explicitly addresses microplanning. We feel that our work provides an optimal path between task-oriented generators (which face problems due to the interrelationships between the tasks) and traditional incremental generation (which does not take advantage of problem decomposition as discussed below).

In what follows we describe our ontology-based modularization, the kind of constraints which can be automatically set up within such a paradigm, and the control mechanism we employ to process it. We focus on the task of lexicalization, but other microplanning tasks have been handled similarly. We conclude with a discussion of the avoidable barriers inherent in most current approaches, along

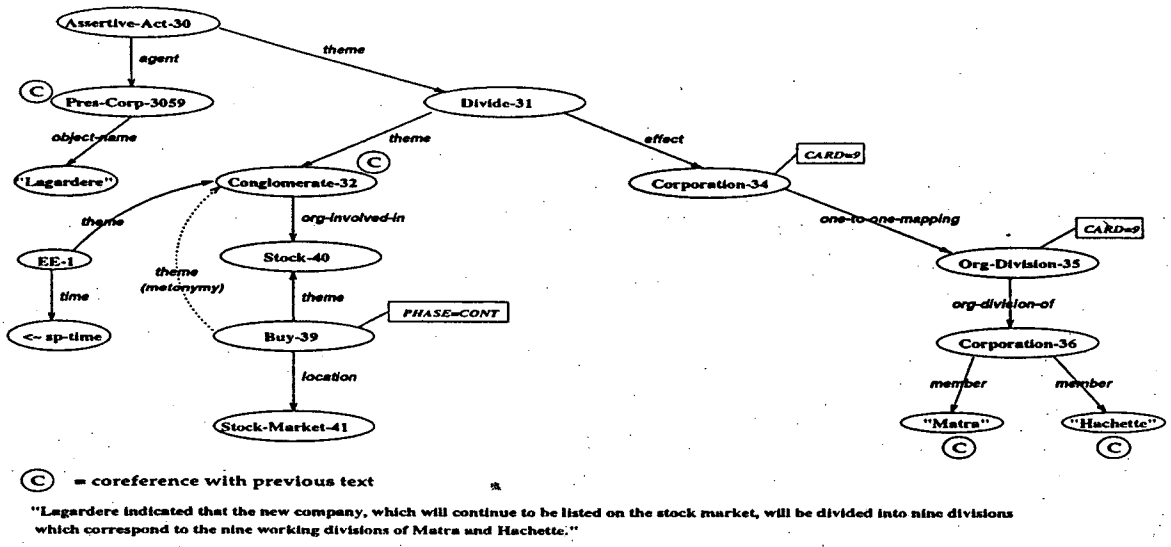

Figure 1: Input Semantic Representation to the Mikrokosmos Generator

with their attempts at circumventing them, and how our approach eliminates many of the problems. We also point out differences between our approach and that of the incremental generators.

2 Ontology-Based Modularization

In contrast to modularization by tasks such as discourse structuring, clause structuring and lexical choice, the Mikrokosmos project (http://crl.nmsu.edu/Research/Projects/mikro/index.html) attempts to modularize on the ontological and linguistic data that serves as inputs to the text generation process, that is, based on the types of inputs we expect, not on the types of processing we need to perform. A typical semantic representation that serves as the input to the generation process is shown in Figure 1. This semantic input was produced by the Mikrokosmos analyzer from an input Spanish text.

The generation lexicon in our approach is essentially the same as the analysis lexicon, but with a different indexing scheme: on ontological concepts instead of NL lexical units[1]. ([Stede1996] is an example of another generator with a comparable lexicon structure, although our work is richer, including collocational constraints, for example). The generation lexicon contains information (such as, for instance, semantics-to-syntax dependency mappings) that drives the generation process, with the help of several dedicated microtheories that deal with issues such as focus and reference (values of which are among the elements of our input representations). The Mikrokosmos Spanish core lexicon is complete with 7000 word senses defined; the English core lexicon is still under development with a projected size of over 10,000 word senses. Both of these core lexicons can be expanded with lexical rules to contain around 30,000 entries ([Viegas et al.1996]).

Lexicon entries in both analysis and generation can be thought of as "objects" or "modules" corresponding to each unit in the input. Such a module has the task of realizing the associated unit, while communicating with other objects around it, if necessary (similar to [De Smedt1990]).

[1]In semantic analysis, the input is a set of words, thus the lexicon is indexed on words. In generation, the input is concepts, so it is indexed on concepts.

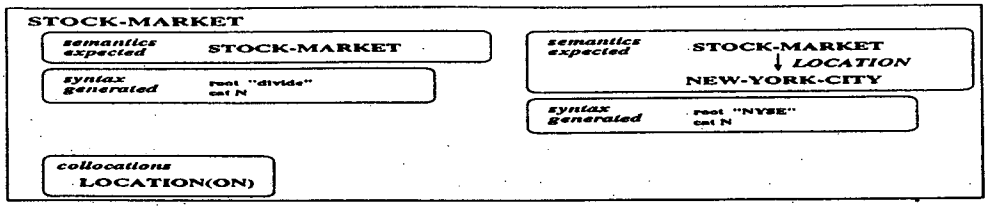

Figure 2: OBJECT Lexicon Entries - A Simplified View

Each module can be involved in carrying out several of the tasks like those listed by Wanner and Hovy. For instance, modules for specific events or properties are used in setting up clause and sentence structures as well as lexical choice, as will be shown below. Interactions and constraints flow freely, with the control mechanism dynamically tracking the connections[2]. One outcome of this division of labor between declarative data and the control architecture is that the bulk of knowledge processing resides in the lexicon, indexed for both analysis and generation. This has greatly simplified knowledge acquisition in general [Nirenburg et al.1996] and made it easier to adapt analysis knowledge sources to generation as well as to convert knowledge sources acquired for one language to use with texts in another.

Below we sketch out how this organization works. We begin by describing the main types of lexicon entries with the goal of demonstrating how each performs various generation tasks. We then take a look at the different types of constraints associated with each kind of entry.

2.1 Types of Lexicon Entries

The main types of lexicon entries correspond to the ontological categories of OBJECTS, EVENTS and PROPERTIES (for simplicity, we will avoid discussion of synonyms and stylistic variations):

- **Objects.** In English, objects are typically realized by nouns, although the actual mapping might be rather complex [Beale and Viegas1996]. In general, object generation lexicon entries can have one-to-one mappings between concept and lexical unit, or can contain additional semantic restrictions, both of which are illustrated in Figure 2. The use of collocational information is described below.
- **Events.** Events, as shown pictorially in Figure 3, can be realized as verbs ("divided") or nouns ("division") in English. Furthermore, the lexicon entries for events typically determine the structure of the nascent clause by mapping the expected case roles into elements of the verb subcategorization frame (or other structures). Rules for planning passives and relative clauses, for instance, are also available. These rules can be used to fulfill different grammatical and clause combination requirements as described below. Conceptually, all the entries produced by these rules can be thought of as being resident in the lexicon. Practically speaking, many of them can be produced on the fly automatically, reducing the strain on knowledge acquisition.
- **Properties.** Properties[3] are perhaps the most interesting of the input types discussed here because they are so flexible. They can be realized as adjectives, relative clauses, complex noun phrases and complete sentences. Often a property is included in the definition of another object or event, such as in Figure 2, where the LOCATION is included in the object entry. CASE-ROLE-RELATIONS

[2]Although our constraint-based planner supports truth maintenance operations, in the "fuzzy" domain of natural language semantics it is often more appropriate to speak of "preferences"

[3]RELATION and ATTRIBUTE are the subtypes of PROPERTY.

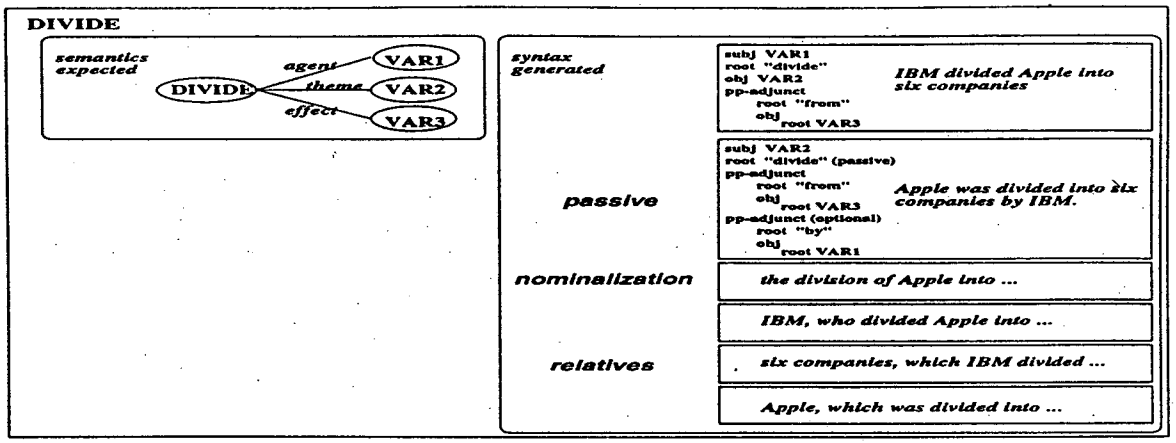

Figure 3: EVENT Lexicon Entries - A Simplified View

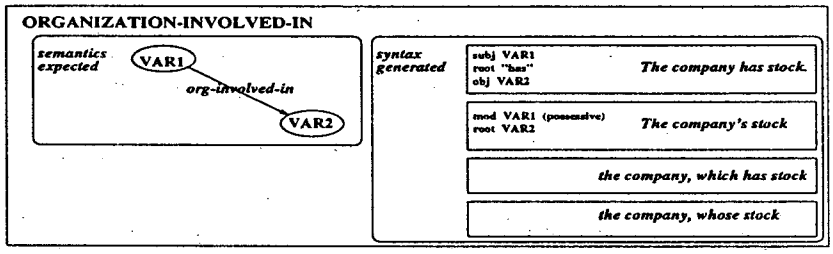

Figure 4: PROPERTY Lexicon Entries - A Simplified View

typically are consumed by the event entry, except in the case of some nominalizations. DISCOURSE-RELATIONS contribute to setting up sentence boundaries, sentence ordering and pronominalization. Figure 4 is an example of RELATION.

2.2 Constraints in Sentence Planning

The above generation lexicon entries are the primary knowledge sources used in the generation process[4]. Five different types of constraints are automatically generated which constrain the combinations of entries allowed to globally realize a semantic input. The Mikrokosmos control mechanism efficiently processes constraints to produce optimal global answers.

Binding Constraints. One of the primary advantages of input-based modularization is that the individual knowledge sources (lexicon entries) can be **grounded** in the input they expect to be matched against. For instance, in Figure 3, the semantic input expected shows three variables, corresponding to the three case roles normally associated with a DIVIDE event. The process of linking these variables to the actual semantic structures for a particular input is known as **binding**.

[4]Due to space limitations, we are glossing over important generation microtheories such as sentence boundary determination and coreference implementation.

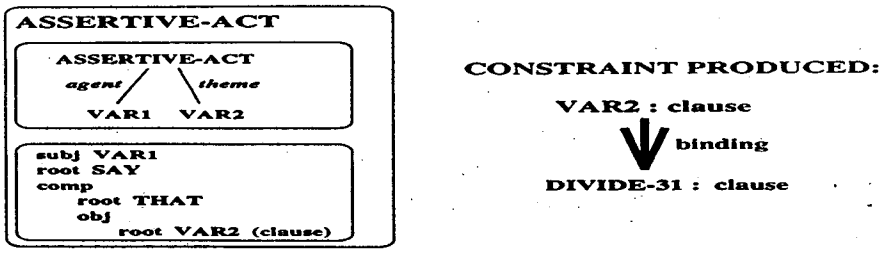

Figure 5: Grammatical Constraints

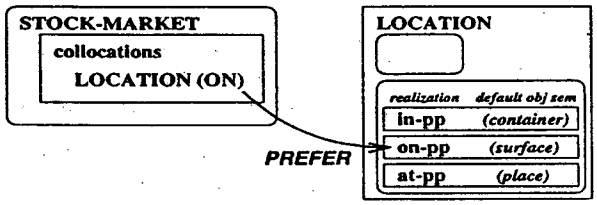

Figure 6: Collocational Constraints in Lexicon Entries

For the input shown in Figure 1, **VAR2** will be bound to CONGLOMERATE-32 and **VAR3** will be bound to CORPORATION-34.

Notice that, for this example, no AGENT exists for the DIVIDE-31 event, so that **VAR1** will be left unbound. Binding constraints will simply eliminate any syntactic choices that contain non-optional unbound variables. In this case, it will rule out the first syntactic realization for DIVIDE shown in Figure 3.

The grounding of the input afforded by the binding process also allows us to simplify the other types of constraints described below. Each of these types of constraints, automatically processed in our system, in task-based systems typically require complex rules to be acquired manually.

Grammatical Constraints. An example of a grammatical constraint is shown in Figure 5. A lexicon entry can specify grammatical constraints on the realization of any of the variables in it. One possible syntactic realization for ASSERTIVE-ACT is shown. It requires its **VAR2** to be realized as a clause. This particular entry allows the system to produce "John said that Bill went to the store" but not "John said that Bill." A comparison with Figure 1 shows that the binding process will link **VAR2** of the ASSERTIVE-ACT entry to DIVIDE-31. In effect, the resulting constraint will eliminate any realization for DIVIDE-31 (in Figure 3) that does not produce a full clause at the top-level, through nominalization and relativization. It should be stressed that this filtering occurs only in conjunction with the given realization of ASSERTIVE-ACT; there may be other realizations that would go fine with, for example, a nominalized realization of DIVIDE.

Collocational Constraints. Figure 6 illustrates the familiar notion of collocational constraints. Again, the fact that the lexicon entry is grounded in the input allows a simple representation of collocations. In this case, the different realizations of LOCATION usually correspond to the semantic type of the object. Collocations can be used to override the default. The co-occurrence zone of the STOCK-MARKET entry simply states that if it is used as the range of a LOCATION

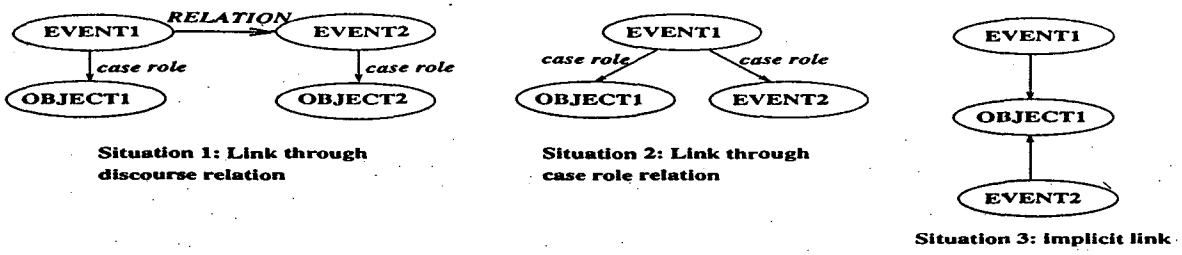

Figure 7: Inputs that May Lead to Clause Combinations

relation, then the LOCATION relation should be introduced with "on." This produces an English collocation such as "the stock is sold **on** the stock market" as opposed to the less natural "... sold at the stock market." Notice that no additional work on collocations needs to be performed beyond the declarative knowledge encoding. The constraint-based control architecture will identify and assign preferences to collocations.

Clause Combination Constraints. Various kinds of constraints arise when clauses are combined to form complex sentences. The strategies for clause combination come from three sources:

- Directly from a lexicon entry associated with an input. For example, a discourse relation such as *CONCESSION* might directly set up the syntax to produce a sentence structure such as "Although Jim admired her reasoning, he rejected her thesis."
- Verbs which take complement clauses as arguments also set up complex sentence structures and impose grammatical constraints (if present) on the individual clause realizations: "John said that *he went to the store*" or "John likes *to play baseball.*"
- Indirectly, from a language-specific source of clause combination techniques (such as relative clause formation or coordination in English).

These three sources correspond to the three input situations depicted in Figure 7. The first two have explicit relations linking two EVENTs. The first (the non-case-role relation) will have a corresponding lexicon entry which directly sets up the sentence structure, along with specific constraints on the individual clauses. The second possibility typically occurs with EVENTs that take complement clauses as case-role arguments. The lexicon entries for these usually will specify the complex clause structure needed. The third situation has no explicit connection in the input; therefore, some sort of language-specific combination strategy must be used to fill the same task.

Even though the latter case appears to be a situation that requires a task-oriented procedure, in reality it is as easy to use general purpose structure constraints along with a declarative representation of possible transformations available. Assuming, for the sake of illustration, that due to some external reason a single sentence realization of two clauses is preferred[5], a general purpose structural constraint prevents two clauses from embedding a single referent into distinct syntactic structures. For instance, 1 and 2 below are grammatical, but 3 is not, because both the clauses try to use "conglomerate" as their subject.

1. The conglomerate, whose stock is sold on the stock market, was divided into nine corporations.

[5]Constraints which might produce such a preference can come from a variety of sources; a common one is the realizations of discourse relations.

2. The conglomerate, which was divided into nine corporations, is sold on the stock market.
3. *The conglomerate was divided into nine corporations is sold on the stock market.

The general purpose constraint will automatically prevent such a realization and trigger the consideration of subordinate clause transformations.

In addition, the examples of clause combination given above and in Figure 7 all contain examples of coreference across clause boundaries. Although coreference realization has its own microtheory that is triggered by instances of coreference in the text, clause combination techniques may interact with it. For instance, the lexicon entry for a RELATION might specify that a pronoun be used in the second clause.

The important thing to note for this presentation is that these types of constraint are either directly found in the lexicon or are produced automatically by the planner. Special situations such as coreference can be easily identified because the lexicon entries are grounded in their inputs. This method appears to be much simpler than those needed by task-based generators.

Semantic Matching Constraints. Matching constraints take into account the fact that, first of all, certain lexicon entries may match multiple elements of the input structure and, secondly, that the matches that do occur may be imperfect or incomplete.

In general, the semantic matcher keeps track of which lexicon entries cover which parts of the input, which require other plans to be used with it, and which have some sort of semantic mismatch with the input. The following sums up the types of mismatches that might be present, each of which receives a different penalty (penalties are tracked by the control mechanism and help determine which combination of realizations is optimal):

- slots present in input that are missing in lexicon entry $->$ undergeneration penalty, plan separately
- extra slots in lexicon entry $->$ overgeneration penalty
- slot filler discrepancies (different, or more or less specific)
  - constant filler values

    HUMAN (age 13-19) - i.e. "teenager" from English input vs.
    HUMAN (age 12-16) - i.e. "age bête" in French lexicon
  - concept fillers

    HUMAN (origin FRANCE) vs.
    HUMAN (origin EUROPE)

A more detailed explanation of these issues is presented in [Beale and Viegas1996]. The important thing to note here is that input-based modularization in our knowledge sources enables this type of constraint to be tracked automatically. In combination with the other constraints described above, we can avoid the complex mechanisms needed by task-based generators for interacting realizations of input semantics.

3 Efficient Constraint-based Processing

The Mikrokosmos project utilizes an efficient, constraint-directed control architecture called *Hunter-Gatherer* (HG). [Beale et al.1996] overviews how it enables semantic analysis to be performed in near linear-time. Its use in generation is quite similar. [Beale1997] describes HG in detail.

Consider Figure 8, a representation of the constraint interactions present in a section of Figure 1. Each label, such as DIVIDE, is realizable by the set of choices specified in the lexicon. Each

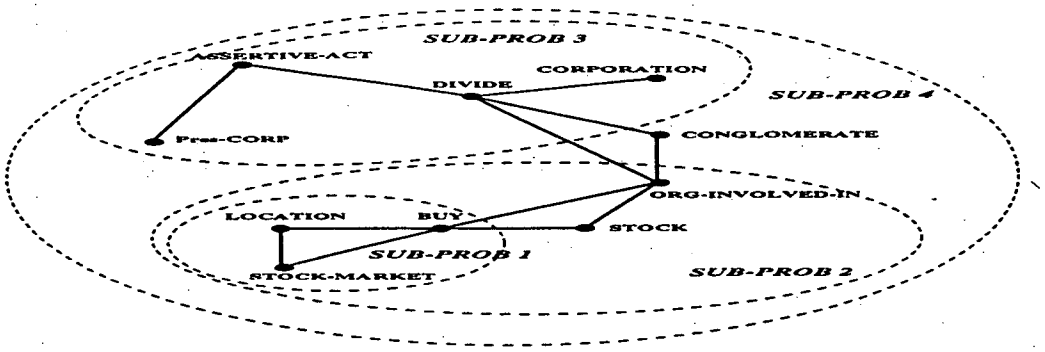

Figure 8: Problem Decomposition

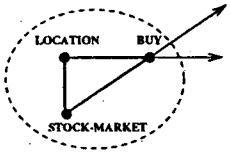

Figure 9: Sub-problem 1

solid line represents an instance of one of the above constraint types. For example, DIVIDE and ORG-INVOLVED-IN are connected because of the structural constraint described above (they both cannot set up a structure which nests the realization of CONGLOMERATE-32 into different subject positions).

The key to the efficient constraint-based planner *Hunter-Gatherer* is its ability to identify constraints and partition the overall problem into relatively independent subproblems. These subproblems are tackled independently and the results are combined using solution synthesis techniques. This "divide-and-conquer" methodology substantially reduces the number of combinations that have to be tested, while guaranteeing an optimal answer. For example, in Figure 8, if we assume that each node had 5 possible choices (a conservative assumption), there would be 5^{10}, or almost 10 million combinations of choices to examine. Using the partitions shown in dotted lines, however, HG only examines 1200 combinations. In general, HG is able to process semantic analysis and generation problems for natural language in near linear-time [Beale et al.1996].

While a detailed explanation of *Hunter-Gatherer* is beyond the scope of this paper, it is fairly easy to explain the source of its power. Consider Figure 9, a single subproblem from Figure 8. The key thing to note is that, of the three nodes, BUY, LOCATION and STOCK-MARKET, only BUY is connected by constraints to entities outside the subproblem. This tells us that by looking only at this subproblem we will not be able to determine the optimal global choice for BUY, since there are constraints we cannot take into account. What we can do, however, is, for each possible choice for BUY, pick the choices for LOCATION and STOCK-MARKET that optimize it. Later, when we combine the results of this subproblem with other subproblems and thus determine which choice for BUY is optimal, we will already have determined the choices for LOCATION and STOCK-MARKET that go best with it.

The following sums up the advantages *Hunter-Gatherer* has for text generation:

- Its knowledge is fully declarative. Note that this is allowed by unification processors [Elhadad et al.1997], but HG gives the added benefits of speed and capability of "fuzzy" constraint processing.
- It allows "exhaustive" enumeration of local combinations.
- It eliminates the need to make early decisions.
- It facilitates interacting constraints, and accepts constraints from any source, while still utilizing modular, declarative knowledge.
- It guarantees optimal answers (as measured by preferences).
- It is very fast (near linear-time).

4 Comparison to Other Generation Systems

Related work exists in two areas: (i) the processing strategy of microplanning tasks, and (ii) the nature and organization of resources used by the microplanner.

There is a strong tendency in generation to deal with microplanning tasks in a small number of modules, which are either structurally or functionally motivated. However, it is recognized that many of the tasks are highly intertwined, so that, in principle, the modules should run in parallel and nearly constantly exchange information. We consider this as a clear hint that a coarse-grained, task-oriented division of microplanning sets up artificial barriers. Repeated efforts of researchers to try and breach those barriers confirm our view.

[Elhadad et al.1997] recognizes that constraints on lexical choice come from a wide variety of sources and are multidirectional, making it difficult to determine a systematic ordering in which they should be taken into account. They propose a backtracking mechanism within a unification framework to overcome the problem. [Rubinoff1992] is perhaps the most strongly focused on this issue. He argues that the accepted division into components "ultimately interferes with some of the decisions necessary in the generation process." He utilizes annotations as a feedback mechanism to provide the planning stages with linguistically relevant knowledge.

Another area of research that belies the unnatural task-based division widely accepted by text generation researchers today is the attempts to control sentence planning tasks. [Nirenburg et al.1989] and more recently, [Wanner and Hovy1996] advocate a blackboard control mechanism, arguing that the order of sentence planning tasks cannot be pre-determined. Behind this difficulty is the reality that different linguistic phenomena have different, unpredictable requirements. Grammatical, stylistic and collocational constraints combine at unexpected times during the various tasks of sentence planning. Blackboard architectures, theoretically, can be used to allow a certain thread of operation to suspend operation until a needed bit of information is available. Unfortunately, in the best case, such an architecture is inefficient and difficult to control. In practice, such systems, as is admitted in both papers above, resort to a "default (processing) sequence for the modules" along with a simplistic truth-maintenance system which ultimately becomes a fail-and-backtrack type of control, completely negating the spirit of the blackboard system. While these shortcomings might eventually be overcome, the fact remains that it was the unnatural division into tasks that necessitated the blackboard processing in the first place.

In this paper, we propose an input data-oriented division of the microplanning task—similar to the way many incremental generators [De Smedt1990, Reithinger1992, Kilger and Finkler1995] divide the task of surface processing. However, the processing of input units as done by the *Hunter-Gatherer* —our microplanning engine—differs significantly from the processing in the incremental generators cited. Thus, an important feature of HG is that it possesses a strategy for dividing the problem of verbalizing a semantic structure into relatively independent subproblems. The

subproblems can be of different size. Into which subproblems the problem is divided depends on constraints that hold between units in the input structure. This strategy greatly contributes to the efficiency of HG. In traditional incremental generators, a unit in the input structure is considered to be a subproblem. Furthermore, HG is bidirectional, i.e., it is usable for both analysis and generation.

References

[Beale and Viegas1996] S. Beale and E. Viegas. 1996. Intelligent planning meets intelligent planners. *Proceedings of the Workshop on Gaps and Bridges: New Directions in Planning and Natural Language Generation, ECAI'96, Budapest*, pages 59–64.

[Beale et al.1996] S. Beale, S. Nirenburg, and K. Mahesh. 1996. Hunter-gatherer: Three search techniques integrated for natural language semantics. *Proc. Thirteenth National Conference on Artificial Intelligence (AAAI96), Portland, Oregon.*

[Beale1997] S. Beale. 1997. Hunter-gatherer: Applying constraint satisfaction, branch-and-bound and solution synthesis to computational semantics. *Ph.D. Diss., Program in Language and Information Technologies, School of Computer Science, Carnegie Mellon University.*

[De Smedt1990] K. De Smedt. 1990. IPF: An Incremental Parallel Formulator. In R. Dale, C.S. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation.* Academic Press.

[Elhadad et al.1997] M. Elhadad, J. Robin, and K. McKeown. 1997. Floating constraints in lexical choice. *Computational Linguistics (2)*, 23:195–239.

[Huang and Fiedler1996] X. Huang and A. Fiedler. 1996. Paraphrasing and aggregating argumentative text using text structure. In *Proc. of the 8th INLG*, Herstmonceux.

[Kilger and Finkler1995] A. Kilger and W. Finkler. 1995. Incremental Generation for Real-Time Applications. Technical Report RR-95-11, DFKI.

[Kilger1997] A. Kilger. 1997. Microplanning in Verbmobil as a Constraint-Satisfaction Problem. In *DFKI Workshop on Generation*, pages 47–53, Saarbrücken.

[Levelt1989] Willem J.M. Levelt. 1989. *Speaking.* The MIT Press, Cambridge, MA.

[Nirenburg et al.1989] S. Nirenburg, V. Lesser, and N. Nyberg. 1989. Controlling a language generation planner. *Proc. of IJCAI-89*, pages 1524–1530.

[Nirenburg et al.1996] S. Nirenburg, S. Beale, S. Helmreich, K. Mahesh, E. Viegas, and R. Zajac. 1996. Two principles and six techniques for rapid mt development. *Proc. of AMTA-96.*

[Panaget1994] F. Panaget. 1994. Using a Textual Representation Level Component in the Context of Discourse or Dialogue Generation. In *Proceedings of the 7th INLG*, Kennebunkport.

[Rambow and Korelsky1992] O. Rambow and T. Korelsky. 1992. Applied text generation. *Applied Conference on Natural Language Processing, Trento, Italy.*

[Reithinger1992] N. Reithinger. 1992. *Eine parallele Architektur zur inkrementellen Generierung multimodaler Dialogbeiträge.* Infix Verlag, St. Augustin.

[Rubinoff1992] R. Rubinoff. 1992. Integrating text planning and linguistic choice by annotating linguistic structures. *Proc. 6th international Workshop on Natural Language Generation, Trento, Italy.*

[Stede1996] M. Stede. 1996. *Lexical Semantics and Knowledge Representation in Multilingual Sentence Generation.* Ph.D. thesis, University of Toronto.

[Viegas et al.1996] E. Viegas, B. Onyshkevych, V. Raskin, and S. Nirenburg. 1996. From *submit* to *submitted* via *submission*: on lexical rules in large-scale lexicon acquisition. *In Proceedings of the 34th Annual meeting of the Association for Computational Linguistics, CA.*

[Wanner and Hovy1996] L. Wanner and E. Hovy. 1996. The healthdoc sentence planner. *Proc. Eighth International Natural Language Generation Workshop (INLG-96).*