

# DEPENDENCY-DIRECTED TEXT PLANNING

Stephen Beale  
Sergei Nirenburg

Computing Research Laboratory  
Box 30001  
New Mexico State University  
Las Cruces, New Mexico 88003

sb@crl.nmsu.edu  
sergei@crl.nmsu.edu  
voice: 505-646-5466  
fax: 505-646-6218

## Abstract

Flexibility and Efficiency. These are the two, conflicting, requirements for a multi-lingual text planning component. Generating a natural language text requires a great amount of flexibility in the rule set employed and, subsequently, in the control mechanism. For any local input semantic structure, there are, in general, many ways to implement it. Each choice produces ripples that move out and affect other parts of the text. A choice that is locally optimal may, in the big picture, constrain text to global choices that are less than optimal. The interplay of constraints, dependencies, and local and global optimization can create a maze of possibilities that must be carefully planned for quality text to be produced in a reasonable amount of time.

Which leads to Efficiency. MLG applications such as database interfaces and, in the future, spoken language interfaces demand speed. In the past, this requirement has often been met by simplifying the semantic input structure as well as the range of language outputs produced. Simplifying semantics leads to misunderstanding and misrepresentation. Simplifying language leads to sterile text at best, and often to incorrect assumptions that again produce misunderstanding.

This paper presents an approach to controlling an MLG text planning component that recognizes and takes advantage of the complex inter-dependencies present in a realistic communication situation. It describes how dependencies can be tracked and then exploited by “opportunistic”, “island driven”, and “sound” processing techniques. Furthermore, it discusses the typology of rules necessary for the text planning component, and how these affect the style of processing.

## KEYWORDS:

Text Planning  
Data Dependencies  
Constraint-Based Processing

## 1. Introduction

We welcome this opportunity to share our experiences in text generation at the IJCAI-95 Workshop on Multilingual Generation. We are excited about many of the issues on the agenda. In particular, our experience in KBMT, particularly the DIOGENES Text Generation project, will give us a unique perspective from which to discuss the questions of IL representation. We hope to be able to participate in the discussions on this and related topics.

In this paper, however, we describe our latest experiment with the mechanism of text planning. The first two questions raised in the conference announcement are addressed in particular:

1. How do requirements for multilingual output delimit possible mechanisms for content selection, text planning and text realization?
2. What should the input to the various levels of MLG look like? <sup>1</sup>

We attempt to demonstrate that, given the complexities of a realistic MLG situation, the system's control mechanism must be able to handle interacting constraints in an efficient manner.

Present text planners can be loosely grouped into one of three categories: hierarchical, systemic, and opportunistic/distributed. Hierarchical planners take an input semantic structure tree and traverse it, creating text plans (or even surface structures) as it goes. If incompatible decisions are detected, optional backtracking occurs (in some systems). The benefit of such processing is its ease in rule writing and processing. The drawback is, of course, backtracking. Incompatible (or non-optimal) decisions are not detected until after they are made and thus must be retracted. In practice, unrestrained backtracking results in unacceptable performance.

Systemic generation [Mann and Matthiessen 83] organizes choices into systems, or categories. Areas such as count, determination, mood, tense, etc., each have their own separate treatment. "Realizers" combine the individual decisions, while the grammar is the overall control mechanism that guides the whole process.

Systemic grammars are an excellent means of describing languages for use in text generation. They provide an explicit framework for discovering and recording the different choices associated with each grammatical feature, along with the reasons behind making one choice over another. On the negative side, interactions between systems are **not** explicitly represented. Realizers are used to combine the results of separate systems, but the systems themselves are kept separate from one another. This may be practical at the level of sentence generation, but planning coherent paragraphs requires a higher degree of interaction.

---

<sup>1</sup>We include language-specific planning rules as a prominent part of MLG inputs.

Opportunistic planners [Nirenburg, et al., 1989] try to take into account expected interactions between rules by planning ahead for them. Before making decisions that ultimately will depend on information not yet available, the opportunistic planner will first try to obtain the information. The benefit to this approach is that backtracking is avoided to a large extent. The main drawback is that it is extremely difficult to predict where and when interactions will occur. This makes the job of controlling the text generator extremely complex, which may (although we continue to work in this area) result in poor performance as well.

Our approach in this project is to try and combine the best of the above approaches while trying to minimize their drawbacks. A hierarchical control strategy is kept for its simplicity and obvious applicability to the tree-structured semantic input.<sup>2</sup> Opportunistic retrieval of information is facilitated by the detailed knowledge of dependencies. Backtracking is largely eliminated by pre-analyzing known dependencies. Systemization, although not yet implemented, will allow us to groups of rules, each directed at one aspect of language, while retaining the ability to let the areas interact in their decisions. In summary, one could call it an island-driven, opportunistic, optimized, sound (need-based) recursive planner. The significance of each of these will be briefly discussed below, and, if time allows at the workshop, concretely demonstrated with specific examples.

## 2. Overview of Project

The following are the highlights of our experiment:

- **GENERATION POTENTIAL:** comparable to Penman and Diogenes 89, and, with extensions, to Diogenes 90. This system is designed to be a general purpose text planner along the lines of Penman and Diogenes 89. With an appropriate lexicon and rule set, it will be able to generate output that is at least as good as those systems. In addition, the opportunistic abilities of Diogenes 90 are also present here. In contrast to these planners, however, this system exhibits the characteristics listed below which enable it to perform more efficiently and, in some cases, produce higher quality output.
- **ISLAND DRIVEN:** takes advantage of “islands of certainty” in input, along with islands created as the planner further constrains problem. Takes advantage of the fact that much of language production is island-driven. By that we mean that within every utterance, whether it is large or small, there are generally “islands” of certainty; places where there is one and only one acceptable choice. Many such islands can be detected even BEFORE processing begins. As a result, certain rules will never be attempted. Rules that produce effects which conflict with preconditions of an island can also be eliminated. In addition to identifying islands before processing, islands that are created due to constraints imposed **during** processing are also detected. Again, their effects are instantiated and rules that lead to conflicts with their preconditions are eliminated.

---

<sup>2</sup>The dependency-based system can be used in a blackboard system such as DIOGENES as well. In fact, it is currently being used in the Mikrokosmos Semantic Analyzer [Nirenburg, Beale, Mahesh, forthcoming].

- **OPPORTUNISTIC:** “looks ahead” for information as needed; prevents needless backtracking. To explain what we mean by “look-ahead”, consider the types of preconditions that may be part of a rule:

1. conditions that relate to the semantics of the immediate input
2. conditions that relate to the semantics of the surrounding text
3. conditions that relate to decisions made at other nodes
4. conditions that relate to actual surface realizations

For instance, a type 1 precondition might ask if the particular semantic structure being processed is a reason-result relation. A type 2 precondition might ask if the current structure is embedded in a reason-result relation. A type 3 precondition might ask whether the parent node is a passive or an active clause. Finally, a type 4 precondition might ask whether the containing clause is greater than 20 words long.

In general, opportunistic planners work on type 2 and 3 preconditions. We have developed a method for identifying and determining whether such preconditions hold before processing begins. This look-ahead procedure can eliminate many possible rule applications before they are ever tried. Type 3 preconditions are systematically tracked to avoid instantiating conflicting rules or rules with impossible preconditions.

- **SOUNDNESS:** our system keeps track of preconditions, automatically detects when a rule’s pre-conditions can no longer be satisfied, automatically detects pre-condition “bashing”. The preconditions for each rule are determined before processing begins. In addition, each rule that could possibly meet that precondition is also calculated. If, during (or before) processing, it can be determined that no more of these “need-fillers” are active for a particular rule’s precondition, then that rule can be pruned out of the search tree, with any resulting new islands treated as described above. This feature relies on a constraint propagation algorithm developed for the system.
- **OPTIMIZED:** rule set dependencies are pre-analyzed to the point where all effects of instantiating a rule are known beforehand, and processing is reduced to a knowledge-free trip through a tree of 0’s and 1’s. The preconditions and effects of each rule that can be applied at a certain node are “compiled” into a very efficient format. For every possible rule application, we can tell immediately, for each node in the input semantic tree, which rules must be eliminated because their effects conflict with the present effect or conflict with the preconditions of the current effect, or because their preconditions conflict with the present effect. In practice, after optimization, we proceed through the “recursive” part of the planner without regard to the actual preconditions and effects. We simply instantiate a rule, which automatically eliminates all other conflicting rules. And keep in mind that every time a rule is eliminated, it is checked to see whether an island was created (in which case its island constraints are instantiated as described above), and whether any other rule’s preconditions have become impossible to meet (in which case the rule in question is removed).
- **MODIFIED RECURSIVE DESCENT ALGORITHM:** takes advantage of above features to eliminate at least 80 percent of the processing in the experiment (compared to basic recursive descent algorithm).

### 3. Brief Discussion of Inputs, Outputs and Processing

#### 3.1. SEMANTIC INPUT

The interlingual text language created for this project is a variation of the one used in DIOGENES and currently being used in the Mikrokosmos project. In general, it is a hierarchical arrangement of propositions linked together into propositional clusters (which in turn can be linked together) by rhetorical relations. The propositional representation relies on an ontology, or language-independent world model. Instances of ontological concepts become elements of text meaning representation - either free-standing entities (frames) or values of other entities' properties (slot fillers). Statements of the event's aspectual and modal properties are also included. Each concept used in the proposition has its place in the ontological hierarchy. Discourse and pragmatics-related information is represented through speaker attitudes to text meaning elements (e.g. expectation, evaluative, etc.) and general stylistic features.

In analysis, using the world knowledge contained in the ontology is extremely important in disambiguation. For instance, knowing that a ball is used in soccer helps disambiguate the sentence "He had a ball" in a soccer context. In generation, world knowledge is used to a much smaller extent. The ontology essentially provides hooks on which lexical items can be hung. There are cases, however, where a concept's ancestry is important; for instance, knowing whether a particular concept is animate may influence the choice of verb. The ontology is also important in generating definite descriptions.

The speaker's intent in communication can also be specified, although, at present, none of our rules make use of such information.<sup>3</sup>

Our experiment assumes a hierarchical backbone, but beyond that would be able to handle a large variety of semantic information. Anything can be attached to one of the nodes, and rules created to handle its occurrences. We are very interested in issues of IL representation and look forward to applying advances in that area to this paradigm.

#### 3.2. SYSTEM OUTPUT

We refer to our work as text planning, as opposed to text realization. Our main goal is to produce lexical choices, determine sentence and paragraph boundaries and establish word and sentence order in a coherent, fluent manner, all in line with the input semantic representation and global pragmatic factors. We refer to our output as "text plans." They contain the information needed by a surface generator to produce the actual surface text.

As we have been primarily concerned with machine translation, the content planner, for us, is the semantic analyzer. Thus, there has been no need to interleave content planning with

---

<sup>3</sup>The relation of rhetorical structure to speaker purpose is an area into which we hope to gain further insight.

text planning, although we recognize the attractiveness of this in other MLG applications. We believe the dependency-directed planning paradigm is applicable to content planning as well; all the more so if it is interleaved with text planning, since interacting constraints are our main focus.

### 3.3. RULE FORMAT

The typology of preconditions and constraints that should be expected is an important area of research. Our preliminary analysis identifies three main types of rules:

- **“DET”** ermine preconditions: those that can be determined by looking at the input semantic structure or global pragmatic values. The simplest of these are the global preconditions, such as checking the level of formality. Many rules can be easily discarded on account of improper pragmatic values. A little more complex are the preconditions that depend on the input semantic structure. For instance, a rule that elides the agent in a proposition following another proposition that has the same agent. If such preconditions can be identified, they can be applied before true processing begins, based solely on the input text.

This is an example of the opportunistic processing which can eliminate much backtracking. If a rule with such a precondition was not eliminated before true processing begins, another rule that depends on the effects of the bogus rule might also be retained. By pruning all improper DET rules before processing, we can accurately assess the dependencies of the remaining rules and eliminate any that cannot be satisfied.

- **“DEP”** end preconditions: those that depend on constraints set by previous or future planner decisions. Planning sentence boundaries is one example of this type of rule. A rule that connects the two propositions in a result-reason relation with “because” obviously cannot also have a sentence boundary placed between them. Thus, the sentence boundary determination depends on the relation implementation, and vice versa. On a slightly more abstract level, if we include a piece of information in one spot, we might be more inclined to elide it, or express it differently, in another.

It is our opinion that these are the types of preconditions that are: 1) the most difficult to identify, and 2) the most important in producing fluent text. We believe they have been largely ignored in the past due to their difficulty, but need to be seriously researched in the future. This project supplies a control mechanism that can handle such preconditions efficiently.

- **“SUR”** face preconditions: those that depend on the state of the surface text. Surface forms have the greatest impact on determining sentence boundaries. The number of words, or even syllables, is an important determinant in deciding where to place sentence boundaries. Generating pronouns is another example. Here, it is useful to keep track of “focus” in a text, in order to help decide which participants can be safely pronominalized. Methods to do this, however, critically rely on the surface ordering of clauses, a SUR precondition.

There are two possible methods to deal with SUR preconditions. First, one can carry along an intermediate surface representation of the text that is being created as the input tree is being traversed. This, however, is extremely costly. We prefer to pre-analyze the dependencies between rules, find the best solution, then apply the effects at the end, rather than applying them at each decision point, retracting any effects when backtracking occurs. A variant on this approach would be to carry along a minimal representation, such as the number of words used, that would solve some of the specific problems.

Perhaps a better approach, and one we found sufficient, is to let the SUR constraints “fall through” to the end of the process. For instance, deciding that a sentence has become too long can be determined after the effects are applied. Backtracking can then be instigated, or, better yet, a “correction” mechanism can be applied to the text. This seems to be more in line with human speech generation techniques, although we are not prepared to support that statement at this time. We also found that pronoun generation, which apparently relies on surface conditions, can be left to a post-processor that acts upon the output text. Surface preconditions are one of the most difficult areas of control that needs to be further investigated.

### 3.4. General Processing

First<sup>4</sup>, all of the rules that may be applicable to a node in the input semantic tree are collected. Those rules whose DET preconditions are not satisfied are eliminated. Next, the dependencies between rules are calculated. For each rule, all of the preconditions that are needed are recorded. The effects of each rule are also recorded and propagated to all possible affected nodes. At this point, it can be determined which rule(s) have effects that satisfy the preconditions of other rule(s). We also determine, for each rule, which other rules conflict with its effects<sup>5</sup>.

Rules that have preconditions that cannot be satisfied at all are eliminated. Whenever a rule is eliminated, all other rules that have preconditions that depend on that rule are re-checked, and eliminated if necessary. When a rule is eliminated, island conditions are also checked: if the node has only one valid rule left<sup>6</sup>, then we know that that rule must be used. We can thus eliminate any rules that have preconditions or effects that conflict with its effects.

At this point, we have eliminated all rules that cannot possibly be applied with the given input. We then begin the modified recursive descent. At each node of the input semantic tree, the first active rule is instantiated<sup>7</sup>. All of its island constraints are implemented,

---

<sup>4</sup>We will go over specific examples in detail at the workshop. Also see [Beale, 94].

<sup>5</sup>For instance, one rule says to make a sentence boundary, another rule has a precondition that there cannot be a sentence boundary

<sup>6</sup>or, when we implement system of rules, if a node has only one valid rule left in a particular system

<sup>7</sup>We plan to implement heuristics for choosing which rule to pick.

effectively failing all the rules that conflict with it. At this point, the soundness and island conditions of each of the nodes that had rules failed is checked and acted upon, which in turn may fail other rules, etc.. When equilibrium is reached, the next node in the tree is visited in a depth-first traversal.

There are two exceptions to this general mode of operation. First, if the node is already an island, nothing needs to be done. This is true because all of the island constraints were implemented at the time the node first became an island<sup>8</sup>. Secondly, if as a result of instantiating the rule (and the subsequent island and soundness processing) a node in the tree has all of its rules failed<sup>9</sup>, then backtracking commences. If there are other rules in this node, they are tried. If all the rules in this node lead to failure, then we backtrack up to the previous node, and try the next rule there.<sup>10</sup>

The modified recursive descent is continued in this manner until a solution is discovered. A score is calculated for it, based on a combination of the scores for the individual rules used. At this point, we can either stop if the rule satisfies a minimum threshold, or attempt to find more answers by artificially instigating backtracking.

It should be noted that backtracking is minimized by the system. The soundness and island processing work to eliminate conflicting sets of rules without the need to try them out, fail, and backtrack. Also, the optimizing techniques identify all of the nodes that must be failed every time a rule is instantiated. Thus, it is not necessary to traverse the tree down to where the failure occurs in order to detect the failure. This eliminates much of the wasted processing associated with backtracking.

#### 4. Conclusion

This discussion is necessarily brief. It is intended to allow us to present our credentials for the workshop. It is clear that the workshop should not deteriorate into a mere sequence of unconnected presentations of new results. It should be devoted to the discussion of the issues listed in the announcement. We intend to use this material as a platform for discussing issues in text planning mechanisms (question 1) as well as the inputs to various levels of MLG (question 2). We are also preparing materials for the discussions on interlingual representation.

---

<sup>8</sup>If the node was originally an island, the island constraints were set during the initial island processing

<sup>9</sup>A condition that is also checked for whenever a rule is failed.

<sup>10</sup>We plan on implementing dependency-directed backtracking, which should be straightforward, given our dependency knowledge.

## Bibliography

Beale, Stephen. 1994. Dependency-Directed Text Generation. Technical Report MCCS-94-272, Computing Research Laboratory, New Mexico State University.

Mann, William and Christian M. Matthiessen. A Demonstration of the Nigel Text Generation Computer Program. In R. Benson and J. Greaves, editors, *Systemic Perspectives on Discourse: Selected Papers from the Ninth International Systemics Workshop*, pages 50-83. Ablex, London, 1985.

Nirenburg, S., Stephen Beale and Mahesh Kavi. 1995. The Mikrokosmos Semantic Analyzer.

Nirenburg, S., V. Lesser and E. Nyberg. 1989. Controlling a Natural Language Generation Planner. Proceedings of IJCAI-89, Detroit.