Controlling a Language Generation Planner

Sergei Nirenburg*
Center for Machine Translation
Carnegie Mellon University
Pittsburgh, PA 15213

Victor Lesser
Computer and
Information Sciences
University of Massachusetts
Amherst, MA 01003

Eric Nyberg
Center for Machine Translation
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

The set of partially interdependent lexical and syntactic decisions that have to be made in the process of natural language generation are best seen as a complex planning and search problem. This paper discusses the phenomena involved in natural language generation planning and argues that a blackboard-type architecture with agenda-style control is more appropriate for this task than a sequential control architecture with backtracking. The blackboard architecture we describe is implemented in the language generator DIOGENES.

1 Introduction

The process of natural language generation starts with the specification of the 'need to communicate/ the propositional goals for a target language text, and a pragmatic profile of the speech situation — knowledge about the speaker, the hearer, the style of communication, etc. (cf. [DiMarco and Hirst, 1988, Hovy, 1987, Hovy, 1988b, McDonald, 1985]). The generator then has to perform the following tasks:

1. *Content Delimitation.* The system must select which of the propositions related to the propositional goals should be overtly realized, and which should be left for the human hearer/reader to infer (e.g., [McKeown, 19851).
2. *Text Structuring.* The system must determine the order of propositions, the boundaries of sentences in the target language text and the nature of discourse connectives among the elements of the target text (cf. [Hovy, 1988a, Mann and Thompson, 1987]).
3. *Lexical Selection.* The system must select open-class lexical units to be used in the target language text (e.g., [Goldman, 1975, Granville, 1983, Jacobs, 1985, Kittredge *et al„* 1988, Nirenburg *et al.,* 1988a, Ward, 1988, Sondheimer *et al.,* 1988]).
4. *Syntactic Selection.* The system must select syntactic structures for the target language clauses (e.g., [Mann, 1983, McDonald, 1983, Meteer *et al.,* 1987]) and perform closed-class lexical selection according to syntactic structure decisions (e.g., [Pustejovsky and Nirenburg, 1987]).
5. *Coreference Treatment.* The system must introduce anaphora, deixis and ellipsis phenomena when appropriate (e.g., [Derr and McKeown, 1984, Sondheimer *et al,* 1988, Werner and Nirenburg, 1988]).
6. *Constituent Ordering.* The system must establish the order of syntactic constituents in a sentence (e.g., [Hovy, 1988a, Kenschaft, 1988]).
7. *Realization.* The system must map from syntactic representations with lexical insertions into surface strings (e.g., [Tomita and Nyberg, 1988]).

Our natural language generator, DIOGENES [Nirenburg *et* a/., 1988a], is designed to account for all of the above tasks but the first one. The design and development of the content delimitation stage received lower priority due to the intended initial application of DIOGENES — machine translation. The *input* to a generator in a knowledge-based MT system is the result of the analysis of a source language text, and takes the form of a set of statements in an intermediate representation language, or *interlingua.* The interlingua text (1LT) already contains the results of the content delimitation stage of generation.

Tasks 2 through 7 above are best interpreted as planning tasks. The treatment of generation as planning was first investigated by Appell [Appelt, 1985], whose generation system was an application of NOAH-style planning [Sacerdoti, 1977]. Hovy [Hovy, 1988b] argues that while content delimitation is best performed in such a manner, a different, data-driven 'restrictive' planner is best suited for the rest of the planning tasks. The reason for this is that Tasks 2 through 6 perform selection out of a set of alternative expressive means, based on dynamically determined heuristics that depend on the state of a number of different planning processes. The search space is complex and dynamic. The search is non-deterministic, since some early decisions have to be re-evaluated based on additional knowledge and constraints obtained as a result of later processing. The knowledge used for search breaks down into several areas specific to the various input meanings and means of expression (see Table 1).

A most promising method for performing this type of complex task is the use of a flexible, data-driven, agenda-controlled environment that has come to be associated with the blackboard style of control (cf., e.g., [Englemore and Morgan, 19881). A version of such a blackboard-type architecture was designed for the DIOGENES planner. This architecture is very general and is expected to accommodate

*This work was partially funded by an IBM Japan / Tokyo Research Laboratory research contract.

future changes in the volume and type of knowledge. This paper describes the DIOGENES planner for tasks 2 through 6 above.

The purpose of the planner is to map the set of input meanings specified in an ILT into a set of target language expressive means, including:

- target language lexical units
- target language syntactic constructions
- the ordering of words and syntactic constructions
- text architectonics - the ordering of clauses and the boundaries of sentences

A typical ILT contains a wide variety of meanings (cf. [Nirenburg *et al.,* 1988a] or [Nirenburg and Carbonell, 1987]), each of which has a set of associated expressive means. The types of input meaning and their typical means of target language realization are illustrated in Table 1.

In the case where the ILT is incomplete, the DIOGENES planner utilizes default strategies. In the absence of a more specific strategy, it will plan a target language clause for every ILT proposition head (typically, a domain event instance) and a target language noun phrase for each domain object instance (role) in the ILT.

Section 2 of this paper presents the architecture of the DIOGENES planner and its components. In Section 3 the knowledge sources used in DIOGENES are described. Section 4 contains an example of a typical control problem in the planner. In Section 5 we report on the status of DIOGENES and directions of future work, and compare DIOGENES to some other NLG planners.

2 The Architecture

The DIOGENES planner architecture is illustrated in Figure 1. The system components include a set of knowledge sources, organized in clusters according to the type of input meanings and expressive means they handle, a set of blackboard spaces, an agenda with its scheduler, and a set of background knowledge repositories.

2.1 The Input and the Output

The input to the system is represented as a semantic network using the FRAMEKIT knowledge representation system [Nyberg, 1988]. It consists of a set of ILT *clauses,* each of which contains both a proposition with its corresponding set of case roles, and a set of non-propositional meanings, such as those of speech act, focus and modality. The most important links among propositions include causal and temporal ones, while relations among ILT clauses include discourse cohesion and time of speech act.

Figure 2 presents a sample input in graphic form (see sample ILTs in [Nirenburg *et al.,* 1988a]).[1]

[1] From the standpoint of expressive power this network is comparable to the one used by the GOSSIP system [Kittredge *et* a*l*., 1988]; There are, however, a number of important differences — for instance, the GOSSIP network does not include discourse cohesion markers, or, being a computational embodiment of the 'Meaning - Text' model [Mel'Cuk, 1974], does not use case role labels, numbering them instead.

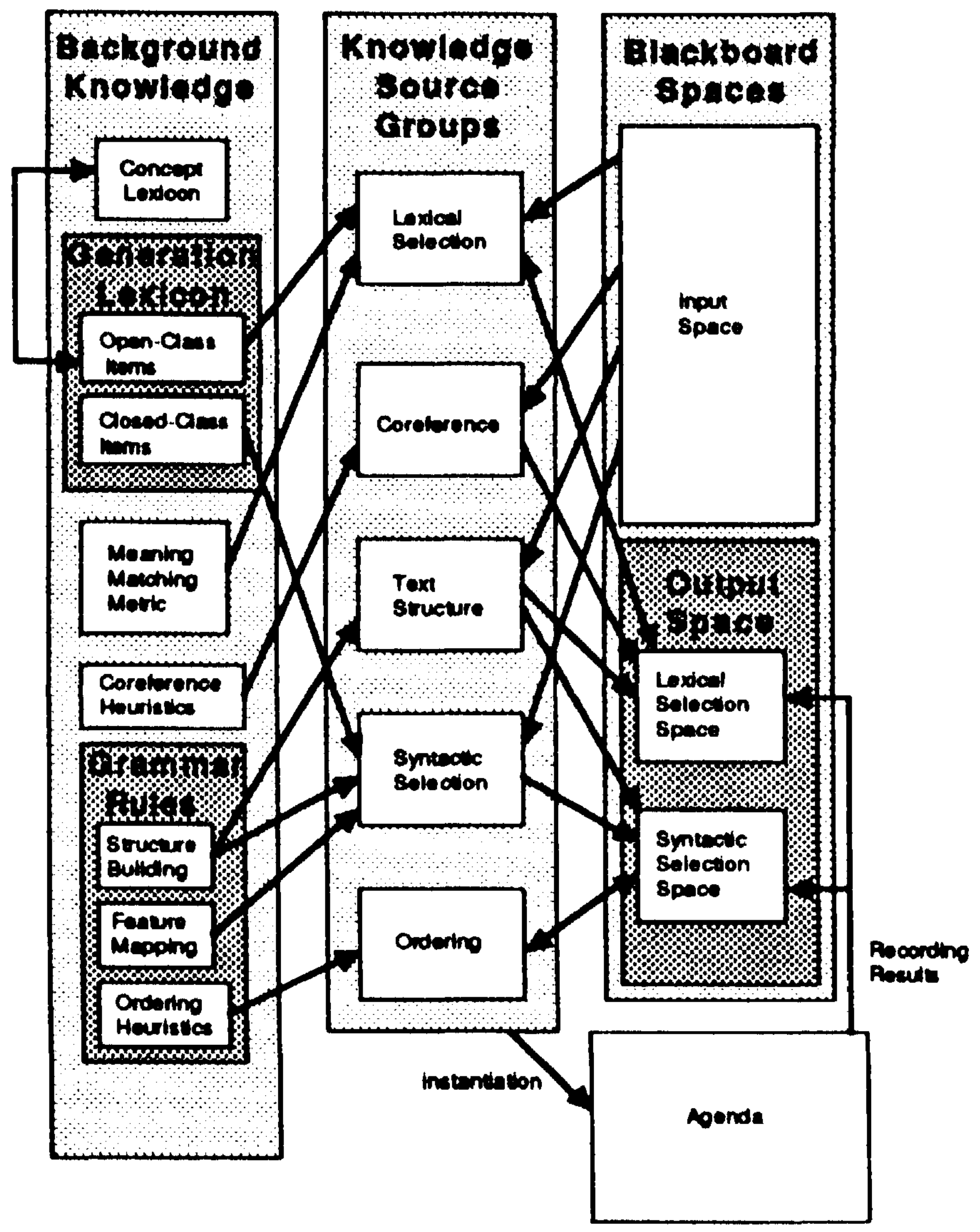

Figure 1: The architecture of the planner in DIOGENES

The *output* of the DIOGENES planner is an ordered set of LFG-like functional structures of target language utterances, with lexical items inserted at the terminal level.

2.2 The Lexicon

The lexicon for natural language generation contains mappings from single units of meaning representation into target language open-class lexical items. The meanings are formulated using an independently constructed model of the domain underlying the texts to be generated. The problems and tasks associated with acquiring a domain model (called the *concept lexicon* in DIOGENES) are described elsewhere [Nirenburg *et al.,* 1988b]. For a detailed discussion of the structure of the lexicon in DIOGENES see fNirenburg and Nirenburg 1988].

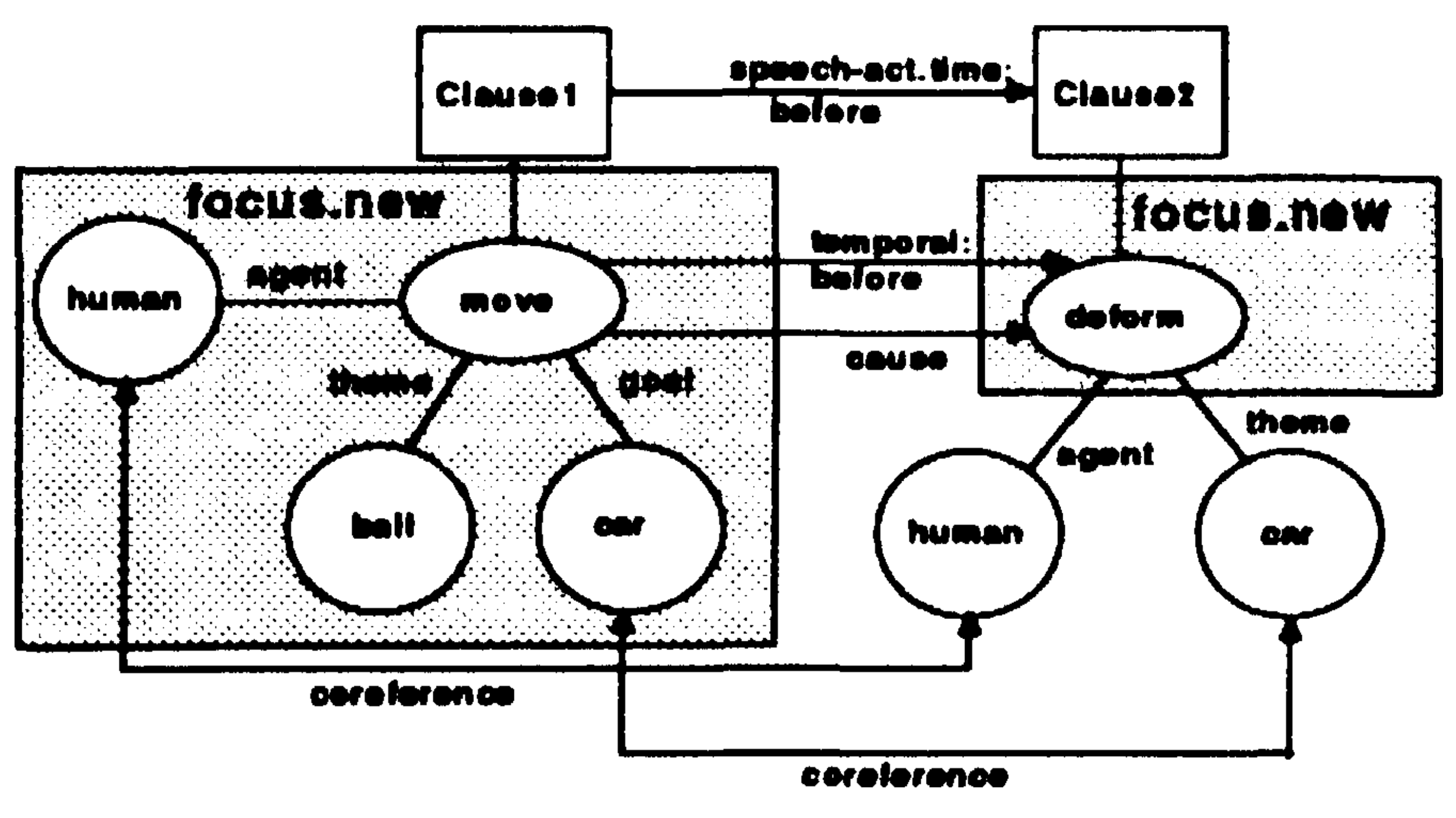

Figure 2: A Sample Input to the DIOGENES planner.

Meaning	Realization Means
Propositional meaning:	
- domain objects, events, and their properties	open-class lexical units
- aspect	open- and closed-class lexical units
- time	closed-class lexical items; tense markers, and sentence and clause order
- causality	closed-class lexical units; sentence and clause order
Pragmatic meaning:	
- speech acts	open- and closed-class lexical units
- focus	lexical units; sentence, clause and word order
- modality	closed-class lexical units
- discourse cohesion	closed-class lexical units; sentence and clause order
Coreferentiality:	use of anaphora, deixis, and ellipsis

Table 1: Meanings and Their Realizations.

3 Knowledge Sources

DIOGENES planning tasks are performed by its knowledge sources (KSs). In this section we describe the current inventory of KSs and illustrate the process flow in the DIOGENES planner. In the following description, the KSs are grouped according to the type of processing they perform. The KSs inside a group tend to be more tightly coupled by the control strategy than those belonging to different groups. However, results posted by KS instances (KSIs) from other groups can also influence the decisions made by a KS, as illustrated below.

3.1 Lexical Selection Knowledge Sources

The task of selecting open-class lexical items (basically, nouns, verbs, adjectives and adverbs) is performed in DIOGENES by a set of three knowledge sources - *GL-Search, Collocationally-Constrain,* and *Select-Best* - and is influenced by the coreference knowledge sources (see below). The lexical selection process in DIOGENES has been described in detail elsewhere (e.g., [Nirenburg and Nirenburg 1988]). Briefly, there are two rounds of GL-Search instantiation. First, a GL-Search KSI is fired for each of the proposition and role heads and when their lexicalization is determined, another round of GL-Search instantiations is fired, if needed, to lexicalize modifiers. The Collocationally-Constrain KSs are fired to ensure contextual collocational compatibility among the lexical selections. Select-Best uses a static proximity metric between an input meaning and each of its potential realizations[2] to select the best match.

3.2 Coreference Knowledge Sources

The knowledge sources in this group are the *Anaphora-KS* and the *Definite-Description-KS.* They are triggered by the existence of coreference finks between roles in the input ILT, and a set of coreference heuristic rules that take into account the lexical selections made for other instances of the coreferential item. If the constraints on instantiation of the Anaphora-KS hold, the instance will generate the pronoun appropriate for referencing a previously generated noun phrase. For a description of this algorithm in DIOGENES see [Werner and Nirenburg, 1988]. If the constraints on instantiation of the Definite-Description-KS hold, the instance will in turn trigger an instance of GL-Search, with the search algorithm modified to create a definite description (i.e., the lexical unit used in the previous reference to the concept will not be included in the candidate list). See [Sondheimer *et al,* 1988] for a discussion of lexical selection for definite descriptions. The coreference links among propositions and among ILT clauses are treated by KSs from the text structure group.

These are stored in the candidate realization list already filtered through collocational constraints.

33 The Text Structure Knowledge Source

The *Text-Structure-KS* determines the ways in which ILT clauses will be distributed among target language sentences. Its instances are triggered for each ILT clause in the input, and appear on the agenda at the same time as those for the GL-Search KS from the lexical selection group. The knowledge that it uses includes discourse cohesion and temporal links among the input ILT clauses. In the absence of this knowledge, the default strategy is to produce a target language sentence for every ILT clause and to retain the order in which the ILT clauses appear in the input (this is recorded in the *time* slot in the *speech-act* subframe of the ILT clause frame). The output of this KS is the skeleton output structure for the entire DIOGENES planner. This structure will then be filled in by the rest of the KSs. Since target language conjunctions and lexical clues for discourse cohesion (e.g., 'therefore', 'moreover') are among the expressive means used by this KS, it can trigger instantiation of a Closed-Class lexical selection KSI (see 3.4 below).

This KS also makes decisions to treat coreference by ellipsis or by using relative clauses. For example, the input

```
(open Johnl bookl)
(read Johnl bookl
      (aspect.phase = begin))
```

would be generated as 'John opened a book and started to read' using the ellipsis strategy, and as 'John started to read the book he had opened* using the relative clause strategy.

3.4 Syntactic Selection Knowledge Sources

There are two main syntactic selection knowledge sources in DIOGENES - *Build-Clause* and *Build-NP,* responsible for producing syntactic structures for target language clauses and noun phrases, respectively. The tasks performed by these KSs include translating semantic relations among ILT propositions and roles (e.g., *agent, theme, goal)* into appropriate

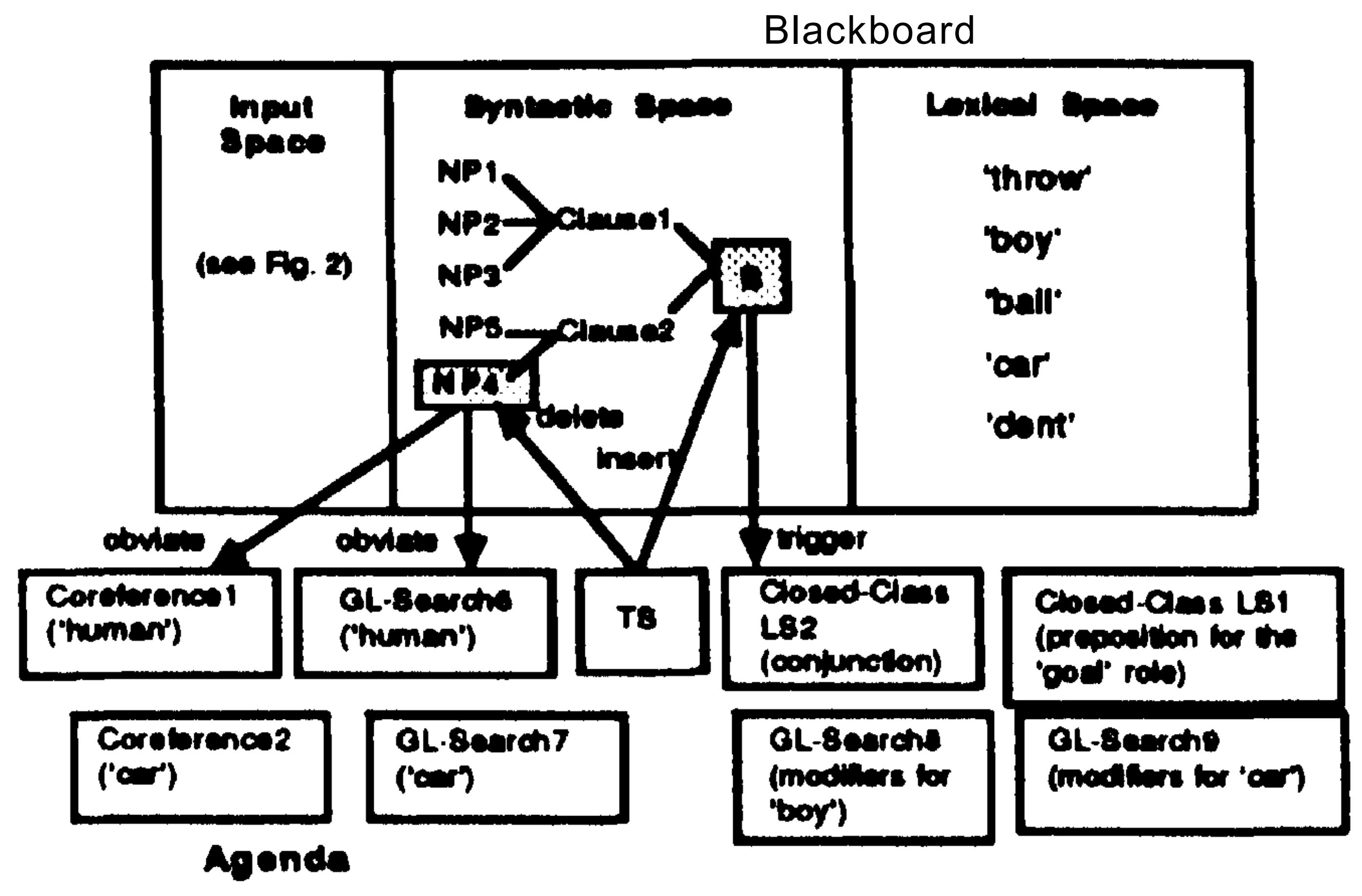

Figure 3: A Snapshot of the DIOGENES Planner. The Ordering KSIs are not shown. The sentence that will be produced is 'A young boy threw a ball at a big car and dented it.'

target language grammatical relations (e.g., *subject, object, adjunct).* During this process, instances of two auxiliary KSs are triggered - *Closed-Class Lexical Selection* KSIs to select closed-class lexical items, such as prepositions or determiners (see [Pustejovsky and Nirenburg, 1987] for a discussion of closed-class lexical selection), and *Feature* KSIs to determine the syntactic features (e.g., *tense, number)* of the output constituents. Build-Clause uses some of the results of the Text Structure KS, e.g., information about whether the current clause has to be generated as a part of a conjoined construction or as a relative clause.

3.5 Constituent Ordering Knowledge Sources

The *Ordering* KS processes constituents in the output structure and stipulates their left-to-right ordering in the output string. At present, the Ordering KS in DIOGENES orders only adjective modifiers of a noun phrase head (see [Kenschaft, 1988] for details). This module will produce, under default conditions, a phrase like 'big black wooden house' and not, say, 'wooden big black house.' The latter *will* be produced if the *focus.given* slot in the ILT pointed to the representation for 'big black house' and the *focus.new* slot, to the representation of 'wooden.'

3.6 Summary and an Illustration

Table 2 illustrates the nature of the DIOGENES knowledge sources. Each KS is shown along with its group, the blackboard events that trigger its instantiation, its inputs and outputs, and the knowledge it uses. This includes static knowledge, such as the lexicons and grammars, and dynamic knowledge, such as the results posted by other KSs on the blackboard.

The task of managing large amounts of knowledge sources has several characteristics with serious implications for control:

1. The KSIs whose results will become a part of the output can be only partially determined *a priori* based on general expectations.
2. Although a large number of KSIs can accumulate on the agenda during the planning process, only a small number will actually have to execute in order for an output to be produced.
3. Since the set of KSIs on the agenda is expected to be large, it is necessary to to a) further limit the search space and b) speed up search by enhancing efficiency through the introduction of extra control knowledge.

In order to enhance cooperation among the KSIs and address these characteristics of the planning process, we introduce the following control strategies:

- Obviation. Since many more KSs may be instantiated than are actually required to produce a solution, a control strategy that involves *obviation* becomes a natural choice. Whenever a particular KSI becomes superfluous because of a particular control decision (as illustrated in Figure 4 below), that KSI may be deactivated on the agenda, so that unnecessary processing may be avoided. Since the obviated KSI is merely deactivated and not removed from the agenda, it need not be re-instantiated if the obviating control decision is retracted later; the obviated KSI can be re-activated simply by re-flagging its status as active. Since obviation is triggered by a contextual situation, its treatment has to be opportunistic.
- A partial ordering on KSI execution. Text-Structure KSIs receive first priority, followed by coreference KSIs with Priority 2. The lexical selection KSIs for proposition and role heads have Priority 3, as well as the syntactic KSIs (Build-Clause and Build-NP). The lexical selection KSIs for modifiers and the Feature KSIs from the Syntactic group are assigned Priority 4. Finally, the Ordering KSIs receive the lowest priority (5).

Figure 3 shows an intermediate state of the system during processing of the input shown in Figure 2. Processing prior to the moment depicted in the figure has completed lexical selection of both the proposition heads and the first four role heads. The results were posted on the Lexical Space and the corresponding KSIs were removed from the agenda (obviated). These events triggered the instantiation of a number of GL-Search KSs for lexical selection of modifiers in the above input components[3]. The system has also built clause and NP structures using two instances of Build-Clause and five instances of Build-NP. The results were posted on the Syntactic Space and the corresponding KSIs were removed. Seven instances of the ordering KS were created (not shown in the figure). An instance of a Closed-Class lexical selection KS was triggered to select the preposition associated with the realization of the 'goal' case role.

Figure 3 also shows the results of a Text-Structure KSI — the two clauses will be combined into a single compound target language sentence. The coordination of clauses will be realized lexically, hence *Closed-Class-LS$_2$* is triggered to select the appropriate conjunction. The agent role of the second clause will be realized through ellipsis; as a result,

[3]Note that no GL-Search KSI is present for finding the modifiers of 'throw', 'ball* and 'dent.* This is because the lexical units that were selected *completely cover* the meanings of the corresponding ILT units.

KS (Group)	Triggered By	Input	Knowledge Used	Output
GL-Search (LS)	ILT, Definite-Description-KS Results	ILT Clauses and Roles	GL, CL; results of Sentence-Structure Build-NP, and Select-Best	CS
Collocationally Constrain (LS)	GL-Search Results	CS	GL, Select-Best results	FCS
Select-Best (LS)	Collocationally-Constrain results	FCS	Meaning Matching Metric	a target language lexical unit
Anaphora-KS (C)	a coreference link in ILT	ILT Role, Select-Best results	Coreference heuristics	a target language pronoun
Definite-Description-KS (C)	a coreference link in ILT	ILT Role, Select-Best results	Coreference heuristics	a set of constraints for GL-Search
Text-Structure (TS)	ILT	ILT Clause, temporal and discourse links	Stylistic rules Grammar rules	Skeleton planner output structure
Build-Clause (SS)	ILT	ILT Clause	Text-Structure KS results, Grammar Rules	clause-level components of output
Build-NP (SS)	ILT	ILT Role; results of Open- and Closed-Class LS and O KSs	Results of C KSs, Grammar Rules	NP-level components of output
Closed-Class LS (SS)	ILT, a new syntactic structure component (result of a SS KS)	Results of LS, SS and TS KSs	Closed-Class GL entries	closed-class lexical units
Feature (SS)	as above	ILT, SS KS results	Feature Mapping Rules	syntactic features in the output
Ordering (O)	as above	unordered set of constituents	Constituent Ordering Heuristics	ordered set of constituents

Table 2: DIOGENES Knowledge Sources. Legend: GL stands for Generation Lexicon; CL, for Concept Lexicon; LS, for Lexical Selection; SS, for Syntactic Selection; C, for Coreference, TS, for Text Structure; CO, for Constituent Ordering; CS for Candidate Set; FCS, for Filtered Candidate Set.

NP4 is *deleted* from the syntactic space, and the KSIs GL-*Searcfo* and *Coreference* are obviated as they are no longer needed.

4 A Sample Control Problem

Control problems arise in our non-sequential architecture when a solution proposed for one planning task influences the outcome of another. Since the architecture does not order the application of KSIs, two KSIs that are triggered as part of the same subtask may create a conflict if the solution choice proposed by one precludes any solution choice by the second. A limited backtracking algorithm, coupled with a simple truth maintenance system, is used to recover from these types of control problems. We will illustrate this algorithm with a sample control problem involving the application of two Collocationally-Constrain KSIs within a single role or proposition.

The initial KSIs of the lexical selection group perform selection of proposition and role *heads,* which, when found, become the system's current choice. Next, for each unexpressed property in the input, the same set of KSs are instantiated, in no particular order.

In general, this is the task of lexical realization in an hierarchically structured context Let us consider the example of two case roles (A and B) within a proposition (C) (cf. Figure 4). Let us assume that the head for the proposition has been selected, and that a lexical unit has already

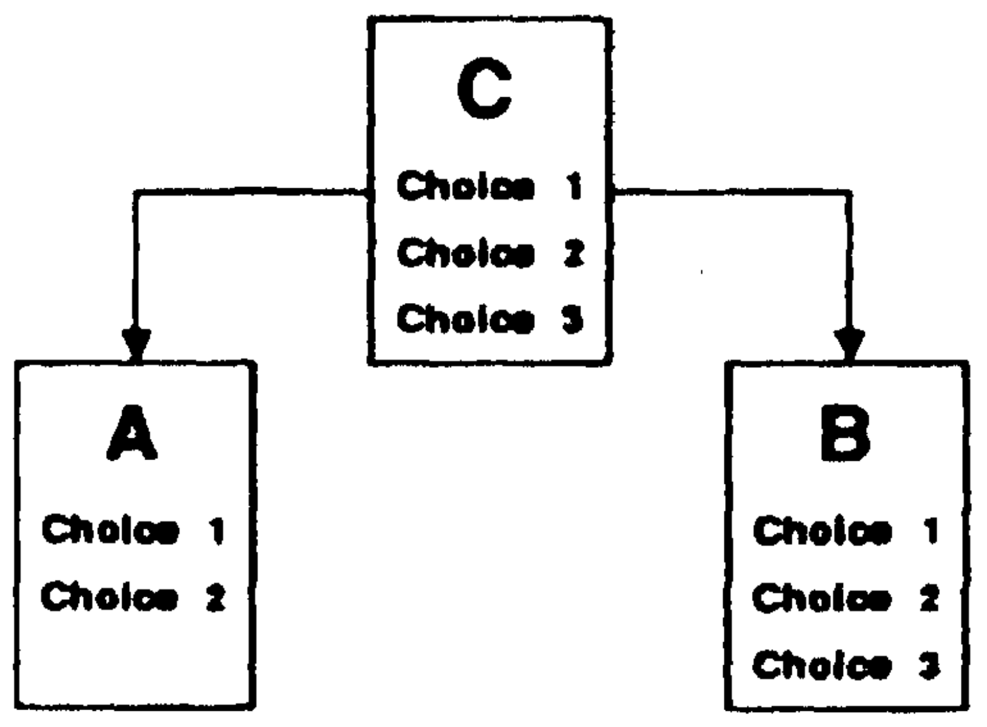

Figure 4: A Realization Problem in an Hierarchical Context

been selected for Role A. Suppose, next, the Collocationally-Constrain KSI responsible for Role B finds that none of its candidates can collocate with the current selection on C. Although this momentary failure is local to a single KSI, it has global consequences — since backtracking will require a different choice on C (to allow a choice of B), all KSIs which assumed the initial choice on C must be re-processed. In other words, one can't assume that the new choice for C will collocate with the selection on A. This type of conflict is resolved in three steps:

1. Detection. A particular KSI reaches a local failure state (in our example, the current choice on C disallows an appropriate choice for B) and this event is posted on an appropriate blackboard space, creating a history of error detection and recovery.

2. Retraction. The decision must be made to ietrigger, in the context of the currently posted history, the KSI whose previous choice must be retracted (in our example, the current choice for C must be retracted). The 'guilty' KSI will then be forced to post its next best choice.

3. Truth Maintenance. Since retraction of a previous solution choice may change the working knowledge of another KSI, any other solution choices that depended on the retracted data must be re-calculated (in our example, the lexeme choice for A must be re-calculated once the next best choice for C has replaced the previous choice).

Our desire is to limit the occurrence of this type of backtracking. In the worst case, an exhaustive search of the space of lexical choices will be made before an acceptable combination or proof of failure is established. Since we initiate backtracking only in extreme circumstances, i.e., when no choices are currently acceptable for a given unit of meaning, extra search is kept to a minimum. In addition, we place an upper resource bound on the number of conflict resolution attempts, after which an arbitrary selection is made with the possible cost of a loss in stylistic quality of the output.

The type of truth maintenance required by our system is simple compared to the complicated TMS components implemented in some systems. Whenever a solution choice is made which, in turn, depends on another choice, the dependent solution adds an IF-ERASED demon to the parent choice, such that retraction of the parent will cause the child to be re-calculated[4]. In terms of our previous example, selecting a choice for A would cause an IF-ERASED demon to be placed on the currently active choice for C. This demon would cause the choice for A to be recalculated in the event that the original choice for C were retracted.

5 Status and Future Improvement

In this paper we have described an initial investigation into the organization of search for natural language generation planning, as implemented (using CommonLisp, the X Windows system and the FRAMEKIT knowledge representation system) in the DIOGENES system (see [Nyberg *et a*/., 1988] for the user's guide). In the next version of the DIOGENES planner we intend to include a more complex control model in the lexical selection phase, means of explicit reasoning about relationships among the tasks and their preconditions, and a more intelligent way of forcing a solution. Certain applications may require the planner to produce multiple solutions instead of outputting the first satisficing choice. This functionality will lead to extra search time at all levels of the blackboard. Finally, we would like to investigate appropriate ways of introducing top-down constraints in our bottom-up search process. These constraints must be introduced not as imperatives, as in schema-based systems [McKeown, 1985] or hierarchical planners [Appelt, 1985], but rather provide additional heuristics for guiding search. This type of strategy has also been advocated by Hovy [Hovy, 1988c], though for different purposes.

A speed-up of the planning process can be achieved by introducing coarse-grain parallelism, whereby different knowledge source classes are treated by different processors. In the current implementation the lexical selection and coreferentiality treatment processes are run in a distributed manner.

We also plan to enhance the expressive means of the planner by including prosodic phenomena (in written text, expressed through the use of italics, boldface, underlining, itemization, etc.) whenever stylistically appropriate.

[4]his is similar to the data dependency maintained in the AMORD system [deKleer *et al.,* 1978], which triggers a search through all data dependency links following the retraction of an assertion.

Acknowledgemen ts

Many thanks to Allan Heydon, Inna Nirenburg, Irene Nirenburg, Victor Raskin, Bob Whitehair, and the current and former members of the DIOGENES project: Scott Huffman, Ed Kenschaft, Rita McCardell and Philip Werner.

References

[Appelt, 1985] Appelt, D. *Planning English Sentences.* Cambridge University Press, 1985.

[deKleer *et a*/., 1978] deKleer, J., J. Doyle, G. L. Steele, and G. J. Sussman. AMORD, A Deductive Procedure System. MIT AI Memo 435, 1978.

[Derr and McKeown, 1984] Derr, M. and K. McKeown. Using Focus to Generate Complex and Simple Sentences. *Proceedings of COLING-84,* 1984.

[DiMarco and Hirst, 1988] DiMarco, C. and G. Hirst. Stylistic Grammars in Language Translation. *Proceedings ofCOLING-88,* 1988.

[Englemore and Morgan, 1988] Englemore, R. and T. Morgan (eds.) *Blackboard Systems.* Addison-Wesley, 1988.

[Goldman, 1975] Goldman, N. Conceptual Generation. In: R. Schank (ed.), *Conceptual Information Processing.* Amsterdam: North Holland, pp. 289-372, 1975.

[Granville, 1983] Granville, R.A. Cohesion in Computer Text Generation: Lexical Substitution. Technical Report MIT/LCS/TR-310, Laboratory for Computer Science, Massachusetts Institute of Technology, 1983.

[Hovy, 1987] Hovy, E. Generating Natural Language under Pragmatic Constraints. Yale University Ph.D. Dissertation, 1987.

[Hovy, 1988a] Hovy, E. Planning Coherent Multisentential Text *Proceedings ofACL-88,* 1988.

[Hovy, 1988b] Hovy, E. Two Types of Planning in Language Generation. *Proceedings of ACL-88,* 1988.

[Hovy, 1988c] Hovy, E. Three Approaches to Planning a Coherent Text. Presentation at the 4th International Workshop on Natural Language Generation, Santa Catalina Island, July, 1988.

[Jacobs, 1985] Jacobs, P. A knowledge-based approach to language production. Ph.D. dissertation, University of California at Berkeley, 1985.

[Kenschaft, 1988] Kenschaft, E. Linearization in DIOGENES. Technical Memo CMU-CMT-88-MEMO. Carnegie Mellon University. June, 1988.

[Kittredge *et a/.*, 1988] Kittiedge, R., L. Iordanskaja and A. Polgaire. Generation using the Meaning - Text Model. *Proceedings of COLING-88,* 1988.

[Mann, 1983] Mann, W. An Overview of the Nigel Text Generation Grammar. USC/ISI Research Report 83-113, 1983.

[Mann and Thompson, 1987] . Mann, W. and S. Thompson. Rhetorical Structure Theory: A Framework for the Analysis of texts. USC/ISI Research Report ISI/RS-87-185, 1987.

[McDonald, 1983] McDonald, D. Natural Language Generation as Computational Problem, in M. Brady and R. Berwick, eds., *Computational Models of Discourse.* MIT Press, 1983.

[McDonald, 1985] McDonald, D. and J. Pustejovsky. A Computational Theory of Prose Style for Natural Language Generation. *Proceedings of European ACL,* 1985.

[McDonald, 1987] McDonald, D., M. Vaughan and J. Pustejovsky. Factors Contributing to Efficiency in Natural Language Generation. In: G. Kempen (ed.) *Natural Language Generation.* Kluwer, 1987.

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

[Mel'cuk, 1974) Mel'cuk, I.A. *Towards a Theory of Linguistic Models of the Meaning-Text Type.* Moscow: Nauka, 1974.

[Meteer *et al.,* 1987] Meteer, M., D. McDonald, S. Anderson, D. Foster, L. Gay, A. Huettncr and P. Sibun. Mumble-86: Design and Implementation. University of Massachusetts Technical Report COINS-87-87, 1987.

[Nirenburg, 1988] Nirenburg, S. Resolution of Lexical Synonymy at the Word Level. Presentation at the 4th International Workshop on Natural Language Generation. Santa Catalina Island, July, 1988.

[Nirenburg and Nirenburg 1988] Nirenburg, S. and I. Nirenburg. A Framework for Lexical Selection in Natural Language Generation. *Proceedings of COLING-88,* 1988.

[Nirenburg *et al.,* 1988a] Nirenburg, S., E. Nyberg, R. McCardell, S. Huffman, E. Kenschaft and I. Nirenburg. Diogenes-88. Technical Report CMU-CMT-88-107. Carnegie-Mellon University. June, 1988.

[Nirenburg *et al.,* 1988b] Nirenburg, S., I. Monarch, and T. Kaufmann. Acquisition of Very Large Knowledge Bases: Methodology, Tools and Applications. Technical Report CMU-CMT-88-108. Carnegie-Mellon University. June, 1988.

[Nirenburg and Carbonell, 1987] Nirenburg, S. and J. Carbonell. Integrating Discourse Pragmatics and Propositional Knowledge in Multilingual Natural Language Processing. *Computers and Translation,* 2, pp. 105-116, 1987.

[Nyberg, 1988] Nyberg, E. The FrameKit User's Guide. Technical Memo CMU-CMT-88-MEMO. Carnegie-Mellon University. February, 1988.

[Nyberg *et al.,* 1988] Nyberg, E., S. Nirenburg and R. McCardell. The DIOGENES User's Guide. Technical Memo CMU-CMT-88-MEMO. Carnegie-Mellon University. August, 1988.

[Pustejovsky and Nirenburg, 1987] Pustejovsky, J. and S. Nirenburg. Lexical Selection in the Process of Language Generation. *Proceedings of ACL-87,* 1987.

[Sacerdoti, 1977] Sacerdoti, E. *The Structure for Plans and Behavior.* Elsevier, 1977.

[Sondheimer *et al.,* 1988] Sondheimer, N., S. Cumming and R. Albano. How to Realize a Concept. Presented at the Workshop on Lexical Semantics, Brandeis University, May, 1988.

[Tomita and Nyberg, 1988] Tomita, M. and E. Nyberg. The Generation Kit and Transformation Kit User's Guide. Technical Memo CMU-CMT-88-MEMO. Carnegie-Mellon University. November, 1988.

[Ward, 1988] Ward, N. Issues in Word Choice. *Proceedings of COLING-88,* 1988.

[Werner and Nirenburg, 1988] Werner, P. and S. Nirenburg. A Specification Language that Supports the Realization of Intersentential Anaphora. *Proceedings of the AAA! Workshop on Natural Language Generation,* 1988.