Managing Sentence Planning Requirements

Eduard Hovy[1] and Leo Wanner[2]

Abstract. In this paper we focus on sentence planning (microplanning) as a distinct phase of the generation process, necessary to compute such aspects as sentence delimitation, sentence-internal organization, reference, etc. Treating each phenomenon as a separate planning task, we illustrate the need for a planning paradigm different from the traditional ones in (at least) this phase because of the complex interactions among these (semi-independent) subtasks. We describe our solution: a blackboard-based sentence planner, in which different subtasks are handled by independent modules that incrementally rewrite and flesh out a text structure (a set of partially specified input statement(s)) to form a fully specified list of sentence specifications.

1 INTRODUCTION

1.1 Planning in generation

It is easy to argue that language generation is at heart a planning task (in contrast to language interpretation, which is at heart the task of inference to the best explanation): Generation originates with one or more communicative goals, and ends with a (more or less) structured list of communication actions (namely, sentences of words). Especially in certain kinds of communications, the planning phase has been modeled as a process of increasing refinement of the original goals (usually quite general in topic), via a structure of intermediate (topically more specified, though still multi-clause) goals, to a linear sequence of specific, clause-level, Speech Act goals.

Traditional Artificial Intelligence planning techniques have been applied to several portions of the generation process, including the single- or two-clause level [Appelt 85] and the multi-clause/paragraph level [Hovy 88b, Moore and Paris 89] and others. However, these applications have generally relied upon strong simplifying assumptions, especially the assumption that the generation process is pipelined (i.e., that 'later' planning tasks such as lexical and clause-internal matters do not affect 'earlier' ones such as content selection and text structure organization).

On the basis of recent work (see below), we believe that the generation process is even more complex than originally viewed, and that this complexity centrally affects the planning procedures. In particular, the pipeline assumption (or its variants; see for example [Reithinger 93]) simply cannot hold for some parts of the generation process; an architecture different from the standard top-down expansion processes is required.

1.2 The need for sentence planning

Automated text generation has often been divided into two major phases: text planning and sentence realization. With the growing numbers of text planners developed over the past decade, whether they were schema application engines [McKeown 85, Paris 87] or text structure planners [Hovy 88a, Moore and Paris 89, Cawsey 89, Maybury 91, Dale 88], has come an increasing recognition of the need for a third, intermediate, phase. This phase, which we call *sentence planning* after [Rambow and Korelsky 92], involves the planning of the contents and organization at roughly the sentence level, distinct from the planning performed during the first phase.

While many early text planners performed all the planning tasks—content selection, text structure planning, and sentence-level planning—together, strong arguments can be made for separating sentence planning tasks from the others.

One argument for separating sentence planning tasks from text planning is that the knowledge required to perform text-level planning differs from that required for sentence-level planning. This argument is analogous to the argument for separating sentence realization (which requires a grammar of the language), from content selection and structuring (which does not). Text-level planning uses knowledge to select content and determine text structure. It ties together intentions and topics by prescribing how an overall communicative intention can be achieved through juxtaposing pieces of content material and other, subordinate, intentions (with, recursively, their associated material). Thus text planning knowledge, captured in schemas or text plans, deals with message *structure* in the development of arguments, explanations, and/or descriptions. It does not specify the actual *phrasing* of the material—this requires information of a different kind.

Lacking this knowledge, text planners usually run out of steam at more or less the sentence level, once the content to be included has been specified. And as any grammar builder knows, a sentence generator with a rich grammar and lexicon can phrase a clause-sized input in literally hundreds of ways! Unless the generator selects locutions at random, an intermediate phase is needed to assemble the detailed grammatical sentence-building instructions required. Only thus can one plug the eloquently named 'generation gap' [Meteer 91].

What knowledge does a sentence planner require? Though it is not concerned with long-range topic juxtapositions, senten-

[1] Information Sciences Institute, University of Southern California, 4676 Admiralty Way, Marina del Rey, CA 90292-6695, U.S.A. tel: +1-310-822-1511 ext. 731; fax: +1-310-823-6714; email: hovy@isi.edu
[2] Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada. tel: +1-519-885-1211 ext. 5344; fax: +1-519-885-120; email: lwanner@logos.uwaterloo.ca

Proceedings of the ECAI 96 Workshop *Gaps and Bridges: New Directions in Planning and Natural Language Generation.*
Edited by K. Jokinen, M. Maybury, M. Zock and I.Zukerman.

ce-level phrasing is obviously a matter of communicative intention. But it also involves grammar and lexis. In fact, as we illustrate below, it is a collection of short-range issues, including sentence delimitation, sentence content organization, and reference expression selection, that require information about local theme, focus, recency of prior mention, style, and so on. By its nature, sentence planning has a much more diverse set of tasks to accomplish than either text planning or sentence realization.

This complexity does not mean the topic has been ignored in past research. Some older systems reasoned with such information. KAMP [Appelt 85], which focused on reference generation (a core aspect of sentence planning), is probably the most sophisticated sentence planner built to date, given the depth and formal complexity of its reasoning. But characteristic of such formal reasoning is its limited scope; KAMP could not produce more than a few variations of two or three specific sentences. Other systems performed a variety of sentence planning tasks—notable among these is DIOGENES [Nirenburg et al. 89] for its sentence planner design, the text and sentence planning of [Horacek 92], the clause phrasing and juxtaposition to achieve bias and stylistic effect in PAULINE [Hovy 88a], and POPEL's planning at the interface between the text planning and realization components [Reithinger 93]. Additionally, sentence planning tasks are listed in [Rambow and Korelsky 92, Hovy 92, Panaget 94, Wanner 94]. [Meteer 91] developed a richly articulated representation formalism in which the output of sentence planners can be captured.[3] And many people have investigated individual aspects of sentence planning, as we mention below.

What has been less studied is the set of all sentence planning tasks and their interaction within a single system.

1.3 Examples of sentence planning tasks

Although the various sentence planning tasks are not unrelated, discussing them individually helps to clarify the issues involved. Recognizing that there are several more, we list the following principal tasks:

Fine-grained discourse structuring. Since the precise boundary between text and sentence planning remains an open question, it is likely that some fine-grained, local, discourse structuring tasks remain for the sentence planner. Issues include the ordering of locally related clauses, and the inclusion or exclusion of explicit discourse markers.

Sentence content delimitation. The sentence planner's task is to apportion the information that must be said into distinct sentences. Little computational research has addressed this question to date, although for work on the determination of temporal, spatial, and causal nuances of predicates see [Pustejovsky 95, Dorr and Olsen 96].

Internal sentence organization. Within a sentence, the sentence planner must allocate the subject, specify the adjuncts (if any), determine the order of preposition phrases, determine the subordination of relative clauses, and so on. These tasks are somewhat interrelated, and are not independent of other high-level sentence planning tasks such as reference and lexical choice. Issues here involve theme and focus [McCoy and Cheng 88, Iordanskaja 92], nuclearity (as used in Rhetorical Structure Theory [Mann and Thompson 88]), salience [McDonald and Conklin 83, Pattabhiraman 82], phrase order [Scott and De Souza 91, DiMarco and Hirst 93, Panaget 94], redundancy [Dalianis and Hovy 93], and so on.

Reference choice. Depending on one's approach, the task of reference can be seen as the central issue in sentence planning (once one has specified how to name each entity and each event, the rest follows) or as a more restricted issue (as endophoric lexical choice that involves just pronominalization and other NP forms). As mentioned above, [Appelt 85] took the wide view; [Dale and Reiter 95] took a somewhat narrower approach; see also [McDonald 78, Tutin and Kittredge 92] for more linguistically motivated approaches. The relative merits of the range of possible positions remain unclear.

Lexical choice. In its narrowest construal, this task involves selecting from a set of semantically equivalent but potentially pragmatically and syntactically different alternatives for entities that have not been mentioned before in the discourse (exophoric lexical choice). Taken broader, it involves both exophoric and endophoric choice. A considerable amount of computational research has been done on various aspects of lexical choice, incl. [Goldman 75, Nirenburg and Nirenburg 88, Hovy 88a, Nogier and Zock 91, Wanner 94].

2 THE SENTENCE PLANNER

2.1 Constraints and desiderata

Given the nature and the current limited state of understanding of sentence planning, a set of relatively general constraints emerge for the design of a sentence planner (SP):

1. The SP must transform an underconstrained input of deep semantics, as given by the text planner, into a suitably constrained output of shallow semantics, as is often required by the realization component.
2. The SP must modularize sentence planning tasks as far as possible, to facilitate design clarity, development, and maintenance. Since the sentence planning tasks are not single-step operations, since they do not have to be performed in strict sequence [De Smedt 90, Nirenburg et al. 89], and since the planner's operation is non-deterministic (that is, early choices may undergo subsequent revision), each sentence planning task should be implemented by a separate module.
3. The intermediate steps of the SP should be accessible and easily interpretable to the builders of the SP, to enable cross-module interconnection and debugging.
4. The SP must be extensible, allowing new modules to be introduced as needed.
5. The level of sophistication of the knowledge within a module must not be constrained by the SP architecture, so that the modules might be crude initially but then can incrementally be refined without impeding throughput. To facilitate this, the rules and knowledge resources employed by the SP modules should be represented as declaratively as possible.

Constraints 1 and 3 suggest that the intermediate form(s) of the data during SP operation be some sort of message-in-emergence; that is, a frame continually evolving from the ab-

[3] Indeed, much has been written about the 'message level': the appropriate 'level' of the input to a sentence realizer, which lies at the heart of determining an appropriate boundary between sentence planning and realization [McDonald and Pustejovsky 85, McDonald et al. 87, Levelt 82].

stract, underspecified input to the more concrete, fully specified final output. One way to achieve this is to see SP modules as tree-transformation engines (viewing the message data structures as trees[4]). This means that their effects can be written as tree-rewriting rules.

Naturally, each module must know which tree-rewriting rule to apply to any given message-in-emergence under given conditions. System networks, as used in Systemic language generation grammars, provide a suitable mechanism. Each module contains a choice network that discriminates arbitrarily finely to a desired state. Each choice point, called a *system*, is associated with two types of information: first, a (set of) *inquiry function*(s) that perform the choice operation (by referring to the system's context: its input, the Reader Model, the Situation Model, the lexicon, etc.); and second, zero or more *realization operators* for each choice alternative that specify the action to be taken, given the current state. These realization operators include tree-rewriting rules, lexicon access functions, and pre-selections that determine later choices in the same or other networks.

Constraints 2, 4, and 5 suggest that the SP have a blackboard architecture. Since we do not know a priori the order of activity of modules or their parts (this is a matter for experimentation), we facilitate message transfer by including a blackboard on which modules can write their results. We have also implemented an Agenda, through which the user can specify the order of activation of modules (in fact, there is no limit on the number of times each module can be run). Thus we use a blackboard for similar reasons to the sentence planner of DIOGENES [Nirenburg et al. 89], although our system differs in important aspects, as will become clear below.

2.2 Sentence planner architecture

A detailed discussion of the architecture of the sentence planner, which reflects the above constraints and desiderata, requires more space than is available here. Briefly, the SP consists of the following modules:

1. A Set of Sentence Planning Task **Modules**: Discourse Structuring, Aggregation, Phrase Ordering (these latter two are part of the Sentence Structuring task), Exophoric Lexical Choice, and Endophoric Lexical Choice.

2. **Knowledge Resources**: the lexicon, the semantic model (split into the Upper and Domain Models), the Reader Model, and the lexicogrammatical resources.

3. The **Blackboards**: the main blackboard (MBB), which contains the latest message expression(s) together with their derivation history, and the control blackboard, which contains bookkeeping information such as the flags that signal the status (running/idle/etc.) of each module.

4. The **Administrator**: the top-level process that invokes modules, updates message expressions, and manages parallel alternative expressions, etc.

5. The **Tree Transformer**: the engine that rewrites (portions of) a message expression as specified by a tree-rewriting rule.

6. The **Network Traverser** (NetMaster): a process that traverses the system network of each module, handles the inquiry choice functions (typically, these criteria pertain either to the input message or to one of the knowledge resources), and upon reaching tree-rewriting rules, hands them off to the Tree Transformer.

[4] Strictly speaking, message expressions and their origins are usually directed acyclic graphs, not trees; this does not affect the design in any way.

2.3 Input and output notation and rewriting

The sentence planner's input is a text specification: a tree whose leaves are written in a notation called *Text Specification Language* (TSL). Its output is a linear sequence of expressions written in *Sentence Planning Language* (SPL [Kasper 89], the input notation used in the sentence generator Penman). TSL resembles SPL in form but ranges in content from deep semantics to shallow semantics (for example, *I have a tummy ache* may in deep semantics be represented by a case frame containing [*:state pain :bodypart stomach*], and in shallow semantics by a case frame containing [*:type generalized-possession :range stomach*], which is closer to its SPL equivalent). This variability enables sentence planning to occur at various levels. We call the sentence planner's intermediate results *pre-SPL expressions*.
Each tree-rewriting rule has the form

[command left-hand-side $\rightarrow$ right-hand-side]

Both left and right hand sides are normal pre-SPL expressions, although they may contain wildcards that match symbols and subexpressions in the normal way. We have implemented the following three commands:

1. ADD: add the statements in the right hand side to any portions of the pre-SPL that matches the left hand side;
2. REWRITE: rewrite any portions of the pre-SPL that match the left hand side as specified by the right hand side;
3. SUPPLANT: replace any portions of the pre-SPL that match the left hand side by the right hand side.

2.4 The planning process

The planning process starts when the Administrator places a TSL fragment onto the MBB and activates the first module on the Agenda.

As mentioned above, it is not linguistically possible to prespecify a fixed sequence in which the modules should always apply. In some cases, mutual dependencies exist between different linguistic phenomena (i.e., planning tasks) that can cause race conditions or deadlock (conflict resolution strategies are discussed in Section 2.5). As default, we define for the Agenda a default sequence of modules that accords with the increasing delicacy of phenomena they address. This default can be overridden at any point by the user.

Upon activation, a module is handed a (possibly fragmentary) pre-SPL expression from the MBB by the Administrator. After operating upon it, the module replaces the resulting pre-SPL(s) back on the MBB. During the operation, pre-SPLs are transformed to form larger and more complete (more detailed) pre-SPLs. The planning process ends when the MBB contains no more pre-SPL(s) to be completed—i.e., when no module can make any further changes to any (pre-)SPL.

Once a module is activated with a pre-SPL fragment, the NetMaster's task is to determine which tree-rewriting rule(s) to apply. It does so by traversing the module's system network, executing its choice point decision routines (the inquiries), and passing any tree-rewriting rules encountered as realizations to the tree transformation engine. This engine matches the left hand side of each rule to the current pre-SPL expression, and, if successful, unifies all variables and transforms the matched portion of the expression to the rule's right hand side. It returns to the NetMaster the newly constituted pre-SPL expression.

The advantages of this architecture flow from the constraints mentioned above. Treating each module as an independent black box allows separate and parallel development by specialists of each sentence planning task. Also, the flexibility of the tree-rewriting rules allows arbitrarily subtle sentence planning operations to be executed. Thus in the worst case, one can for a module write a single rule that simply maps input forms directly to their appropriate output forms, without performing any real planning. Other modules can then still be developed, and the SP continues to run the same way when this crude module is later replaced by a module that executes the same overall transformation via a series of detailed planning steps.

2.5 Inter-module conflict resolution

If a module is activated but is unable to complete processing because it cannot make all the decisions in the network, or if it makes decisions that are known to be possibly incompatible with those made by other modules later on, there are three options for how to proceed: (1) the module must suspend operation until all information required is available; (2) the module must make decisions somewhat blindly and support backtracking when a decision turns out later to be incompatible; or (3) the module must not make the decision but produce all its alternatives in parallel, to be winnowed out by later processing.

We do not discuss the first two options since they are standard AI search techniques. The third option is inspired by the treatment of alternatives in statistics-based NLP. In this option, we allow a module to replace an input with *multiple* alternative outputs instead of only one when it cannot make a choice. All such alternative pre-SPLs are spawned and propagated onto the blackboard, so that other modules can work on them all, as parallel alternatives. Should one of the alternatives later turn out to be incompatible with some module's decision, that alternative simply 'dies' and is removed from the MBB. If, on the other hand, all alternatives remain viable to the end, then the SP has produced more than one valid locution from the input. This option is only feasible if the networks of the modules do not lead to a large number of alternatives. In our model, we have chosen options 2 and 3 as possible modes, although at time of writing the implementation of option 3 remains incomplete.

This design means that on the blackboard at any time contains one or more parallel pre-SPL alternatives, all derived from the (single) original input TSL expression. When a next module is activated, it operates upon *all* the alternatives, taking them all through its networks one at a time, and produces new, better-specified, versions for them all, naturally excepting those that have been pruned out by incompatibilities with the new module's knowledge. Thus one can expect the blackboard to contain, initially, just a single pre-SPL expression, which may gradually multiply into several pre-SPL expressions in parallel, possibly decrease again in number, and eventually end up being one or more viable SPL expressions.

3 SOME EXAMPLES

In this section we illustrate the operation of some sentence planner modules upon the following paragraph, taken from an education brochure for a patient with diabetes, in which we have numbered sections for reference:

> *(0) In a person without diabetes the digestive system breaks some food down into glucose. (1) The blood carries the glucose throughout the body, (2) causing blood glucose levels to rise. (3) In response to this rise the pancreas, (4) a large gland found behind the stomach, (5) releases the correct amount of the hormone insulin into the bloodstream (6) to signal the body tissues to metabolize or burn the glucose for fuel, (7) causing blood glucose levels to return to normal.*

Without changing the meaning in any substantial way, it is possible to rephrase this paragraph many times by varying the expressive forms used. Several classes of expressive form can be identified; each major class we identify with a separate sentence planning task.

Additional examples in another domain can be found in [Wanner and Hovy 96].

3.1 Fine-grain discourse structuring

The discourse structuring module determines the communication of discourse relations. Currently, three major distinctions are made: *explicit/implicit* (for example, the relation *Cause* can be communicated explicitly by a cue word or an appropriate verb such as *to cause*, or implicitly, to be obtained via inference; *nucleus/satellite salience* (for example, salience in *Cause* can be changed by inverting the order of the cause and result); and *cue word selection* (for example, *Cause* can be marked by *because* or *since*).

In the original text, unit (3) reads as follows:

> *[... (2) causing blood glucose levels to rise.] (3) In response to this rise the pancreas, [(4) ... (5) releases the correct amount of the hormone insulin...]*

which is represented as (portions represented in square brackets are abbreviated for reading convenience):

```
(REL1 / CAUSE
      :cause (RISE1 / INCREASE
                      :amount [glucose level])
      :result (REL2 / RELEASE
                      :actor [pancreas]
                      :actee [insulin]))
```

This can be transformed to:

```
(REL1 / RST-RESULT
      :range (REL2 / RELEASE
                     :actor [pancreas]
                     :actee [insulin]))
```

which reads as:

> *[... (2) causing blood glucose levels to rise.] (3) As a result, the pancreas, [(4) ... (5) releases the correct amount of the hormone insulin...]*

One tree-rewriting rule is required. The left hand side matches the input pre-SPL expression; variables are bound (variables are preceded by '?'); and the matched portion is replaced by the right hand side:

```
[SUPPLANT  [(?X / CAUSE          [(?X / RST-RESULT
               :cause ?Y     →        :range ?Z)] ]
               :result ?Z)]
```

This rule has the effect of replacing the (semantic) relation *Cause* with the discourse structure relation *RSP-Result*, and drops the antecedent. Another rule might include an explicit relation cue word:

```
[ADD  [(?X / RST-RESULT    →   [(:lex "as a result")] ]
          :range ?Z)]
```

3.2 Sentence content delimitation

The Content Delimitation module determines the material to be included in each distinct pre-SPL. At present, we include in this task allocating predications to sentences, selecting each predication's constituents, and selecting additional causal, temporal, and aspectual information.
Example units (2) and (3) in the original read:

(1) The blood carries the glucose throughout the body, (2) causing blood glucose levels to rise. (3) In response to this rise...

which is represented as:

```
((REL1 / CAUSE
       :cause (T1 / TRANSPORT
                  :actor [blood]
                  :actee [glucose]
                  :location [the body])
       :result (RISE1 / INCREASE
                      :amount [glucose level]))
 [(3)])
```

This can be transformed to two pre-SPL fragments:

```
(T1 / TRANSPORT
   :actor [blood]
   :actee [glucose]
   :location [the body])
(REL1 / CAUSE
      :cause (RISE1 / INCREASE
                    :amount [glucose level])
      :result [(3)])
```

which read as:

... (1) The blood carries the glucose throughout the body. The (2) resulting rise in blood glucose levels (3) makes...

The required tree rewriting rule is:

```
[REWRITE  [(?X / CAUSE          [?Y
              :cause ?Y     →    (?X / CAUSE
              :result ?Z)             :cause ?Z
           ?W]                        :result ?W)] ]
```

Space limitations prevent the description of the other modules' operations.

4 THE SETTING: HEALTHDOC

The sentence planner described here is being built in the HealthDoc project [DiMarco et al. 95], which aims to develop systems that produce customized health-education material. In view of the complexity of medical issues and the wide range of variation of patient background, education, and situation, medical information brochures are most effective when they are targeted to the patient according to patient-specific features (age, reading level, etc.). Creating suitably customized documents for all patients, however, is a very difficult task, since hundreds or even thousands of variations are possible. HealthDoc combines existing generation technology, such as the sentence generator KPML[5][Bateman 95], with new experimental prototypes, including the sentence planner described here [DiMarco et al. 96].

HealthDoc is being constructed at the University of Waterloo. The sentence planner has been designed and constructed at USC/ISI and at the University of Waterloo. Current work at both sites focuses on the extension of the modules for Discourse Structuring, Aggregation, Ordering, and Endophoric (Reference) Choice, and on the design of new ones.

5 CONCLUSION

Despite previous and current text planning work, we are skeptical that straightforward applications of AI techniques will suffice for planning in NLG in general. The phenomena are too interwoven; the problems are too complex. Although past attempts to factor out various aspects of the generation process and to focus on them individually have achieved some successes (notably in the areas of surface realization, content selection, and paragraph-length text structuring), the more tightly interwoven subtasks of sentence planning remain to be dealt with. Naturally, they can (and should!) be studied in isolation, as in the above-referenced work. But it is important also to study them in concert, in an end-to-end generation system, precisely in order to determine their interdependencies and their interactions.

Given the complexity of the problems, we believe that an architecture such as the one we outline here provides a useful vehicle for research, especially since its modular design allows one to focus on some aspects of sentence planning, while using other peoples' modules to manage others. The ability to vary the order of module application, and to allow a module to produce multiple alternatives in the case of underspecified input, are two interesting new extensions to planning in NLG. We hope that the sentence planner provides a promising tool for investigating several interesting aspects of NLG planning in the near future.

ACKNOWLEDGEMENTS

We would like to thank in particular Bruce Jakeway, as well as all the other members of the Sentence Planning Group of the HealthDoc project: Chrysanne DiMarco, Phil Edmonds, Daniel Marcu, and John Wilkinson.

REFERENCES

Appelt 85 Appelt, D.E. 1985. *Planning Natural Language Utterances.* Cambridge: Cambridge University Press.

[5] KPML stands for *Komet/Penman MultiLingual* and is a development of the Penman system [Penman Group 89].

Bateman 95 Bateman, J.A. 1995. KPML: The KOMET–Penman Multilingual Linguistic Resource Development Environment. *Proceedings of the 5th European Workshop on Natural Language Generation*, Leiden, 219–222.

Cawsey 89 Cawsey, A. 1989. *Generating Explanatory Discourse: A Plan-Based Approach.* Ph.D. dissertation, U. of Edinburgh.

Dale 88 Dale, R. 1988. *Generating Referring Expressions in a Domain of Objects and Processes.* Ph.D. dissertation, University of Edinburgh.

Dale and Reiter 95 Dale, R. and E. Reiter. 1995. Computational Interpretation of the Gricean Maxims in the Generation of Referring Expressions. *Cognitive Science* 19(2):233–263.

Dalianis and Hovy 93 Dalianis, H. and E.H. Hovy. 1993. Aggregation in Natural Language Generation. In G. Adorni and M. Zock, editors, *Trends in Natural Language Generation: An Artificial Intelligence Perspective.* Berlin and Heidelberg: Springer Verlag.

De Smedt 90 De Smedt, K. 1990. IPF: An Incremental Parallel Formulator. In R. Dale, C. S. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation.* Academic Press.

DiMarco and Hirst 93 DiMarco, C. and G. Hirst. 1993. A Computational Theory of Goal-Directed Style in Syntax. *Computational Linguistics* 19(3):451–499.

DiMarco et al. 95 DiMarco, C., G. Hirst, L. Wanner, and J. Wilkinson. 1995. HealthDoc: Customizing Patient Information and Health Education by Medical Condition and Personal Characteristics. In *Proceedings of the Workshop on AI in Patient Education*, Glasgow.

DiMarco et al. 96 DiMarco, C., G. Hirst, and E.H. Hovy. 1996. 'Rewriting is easier than writing': Generation by Selection and Repair in the HealthDoc Project. Submitted.

Dorr and Olsen 96 Dorr, B.J. and M.B. Olsen. 1996. Multilingual Generation: The Role of Telicity in Knowledge Organization and Lexical Choice. To appear in *Machine Translation.*

Goldman 75 Goldman, N. 1975. Conceptual Generation. In R.C. Schank (ed) *Conceptual Information Processing.* Amsterdam: North-Holland Publishing Company.

Horacek 92 Horacek, H. 1992. An Integrated View of Text Planning. In *Proceedings of the 6th International Workshop on Natural Language Generation*, Trento. Springer-Verlag.

Hovy 88a Hovy, E.H. 1988a. *Generating Natural Language under Pragmatic Constraints.* Hillsdale, NJ: Lawrence Erlbaum.

Hovy 88b Hovy, E.H. 1988b. Planning Coherent Multisentential Text. In *Proceedings of the 26th ACL Conference*, Buffalo NY, 163–167.

Hovy 92 Hovy, E.H. 1992. Sentence Planning Requirements for Automated Explanation Generation. In *Proceedings of the Workshop on Explanation Facilities for Model-Based Expert Systems, DIAMOD-Bericht no. 23*, Sankt Augustin.

Iordanskaja 92 Iordanskaja, L.N. 1992. Communicative Structure and its Use during Text Generation. In *International Forum on Information and Documentation* 17(2):15–27.

Kasper 89 Kasper, R.T. 1989. Sentence Planning Language. Unpublished technical document, USC Information Sciences Institute.

Levelt 82 Levelt, W.J.M. 1989. *Speaking.* Cambridge: MIT Press.

Mann and Thompson 88 Mann, W.C. and S.A. Thompson. 1987. Rhetorical Structure Theory: A Theory of Text Organization. In Polanyi, L. (ed) *The Structure of Discourse.* Norwood, NJ: Ablex Publishing Corporation.

Maybury 91 Maybury, M.T. 1990. *Planning Multisentential English Text using Communicative Acts.* Ph.D. dissertation, Cambridge University.

McCoy and Cheng 88 McCoy, K.F. and J. Cheng. 1990. Focus of Attention: Constraining what can be Said Next. In Paris, C.L., Swartout, W.R. and Mann, W.C. (eds) *Natural Language Generation in Artificial Intelligence and Compuational Linguistics.* Dordrecht: Kluwer.

McDonald 78 McDonald, D.D. Subsequent References: Syntactic and Rhetorical Constraints. In *Theoretical Issues in Natural Language Processing-2 (TINLAP).* New York NY: ACM.

McDonald and Pustejovsky 85 McDonald, D.D. and J. Pustejovsky. 1985. Description-Directed Natural Language Generation. In *Proceedings of the 9th IJCAI Conference*, Los Angeles. 799–805.

McDonald and Conklin 83 McDonald, D.D. and J.E. Conklin. 1983. Salience as Simplifying Metaphor for Natural Language Generation. In *Proceedings of the 3rd AAAI Conference.* Washington.

McDonald et al. 87 McDonald, D.D., M.W. Meteer, and J. Pustejovsky. 1987. Factors Contributing to Efficiency in Natural Language Generation. In Kempen, G. (ed) *Natural Language Generation: Recent Advances in Artificial Intelligence, Psychology, and Linguistics.* Dordrecht: Kluwer.

McKeown 85 McKeown, K.R. 1985. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text.* Cambridge: Cambridge University Press.

Meteer 91 Meteer, M.W. 1991. Bridging the Generation Gap Between Text Planning and Linguistic Realization. *Computational Intelligence* 7(4):296–304.

Moore and Paris 89 Moore, J.D. and C.L. Paris. 1989. Planning Text for Advisory Dialogues. In *Proceedings of the 27th Annual Meeting of the ACL.* Vancouver BC.

Nirenburg et al. 89 Nirenburg, S., V. Lesser, and E. Nyberg. 1989. Controlling a Language Generation Planner. In *Proceedings of IJCAI* Detroit MI, 1524–1530.

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

Nogier and Zock 91 Nogier, J-F. and M. Zock. 1991. Lexical Choice as Pattern Matching. In Nagle, Nagle, and Eklund (eds.) *Current Directions in Conceptual Structures Research.* Ablex, NJ.

Panaget 94 Panaget, F. 1994. Using a Textual Representation Level Component in the Context of Discourse or Dialogue Generation. In *Proceedings of the 7th International Workshop on Natural Language Generation*, Kennebunkport ME.

Paris 87 Paris, C.L. 1987. *The Use of Explicit User Models in Text Generation: Tailoring to a User's Level of Expertise.* Ph.D. dissertation, Columbia University.

Pattabhiraman 82 Pattabhiraman, T. 1993. *Aspects of Salience in Natural Language Generation.* Ph.D. dissertation, Simon Fraser University.

Penman Group 89 Penman Natural Language Group. 1989. The Penman Primer, The Penman User Guide, and The Penman Reference Manual. Marina del Rey, CA: Information Sciences Institute, University of Southern California.

Pustejovsky 95 Pustejovsky, J. 1995. *The Generative Lexicon.* Cambridge MA: MIT Press.

Rambow and Korelsky 92 Rambow, O. and T. Korelsky. 1992. Applied Text Generation. In *Applied Conference on Natural Language Processing*, Trento.

Reithinger 93 Reithinger, N. 1993. *Eine parallele Architektur zur inkrementellen Generierung multimodaler Dialogbeiträge.* Ph.D. dissertation, University of the Saarland.

Scott and De Souza 91 Scott, D.R. and C. S. de Souza. 1991. Getting the Message Across in RST-Based Generation. In R. Dale, C. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation* London: Academic Press. 47–73.

Tutin and Kittredge 92 Tutin, A. and R. Kittredge. 1992. Lexical Choice in Context: Generating Procedural Texts. In *Proceedings of COLING 92*, 763–769.

Wanner 94 Wanner, L. 1994. Building Another Bridge over the Generation Gap. In *Proceedings of the 7th International Workshop on Natural Language Generation*, Kennebunkport ME.

Wanner and Hovy 96 Wanner, L. and E.H. Hovy. 1996. The HealthDoc Sentence Planner. In *Proceedings of the 8th International Workshop on Natural Language Generation*, Brighton.