Adapting MUMBLE: Experience with Natural Language Generation

Robert Rubinoff
Computer and Information Science Department
Moore School of Electrical Engineering
University of Pennsylvania
Philadelphia, PA 19104
April 24, 1986

Abstract

This paper describes the construction of a MUMBLE-based [McDonald 83b] tactical component for the TEXT text generation system [McKeown 85]. This new component, which produces fluent English sentences from the sequence of structured message units output from TEXT's strategic component, has produced a 60-fold speed-up in sentence production. Adapting MUMBLE required work on each of the three parts of the MUMBLE framework: the interpreter, the grammar, and the dictionary. It also provided some insight into the organization of the generation process and the consequences of MUMBLE's commitment to a deterministic model.

Track: Engineering
Topic: Natural Language Generation

1 TEXT's Message Vocabulary

The TEXT system [McKeown 85] is designed to answer questions about the structure of a database. It is organized into two relatively independent components: a strategic component which selects and organizes the relevant information into a discourse structure, and a tactical component which produces actual English sentences from the strategic component's output. The original tactical component [Bossie 81] used a functional grammar [Kay 79]; it is this component that has been replaced.[1]

A tactical component for TEXT must be tailored to the form in which TEXT's strategic component organizes information. The strategic component responds to a query with a list of rhetorical propositions. A rhetorical proposition indicates some information about the database and the rhetorical function the information TEXT intends it to perform. For example, the rhetorical proposition:

```
(identification GUIDED PROJECTILE
                (restrictive (TRAVEL-MEANS SELF-PROPELLED))
                (non-restrictive (ROLE PROJECTED-OBJECT)))
```

indicates that TEXT wants to identify guided missiles by saying that they are projectiles and that they have certain attributes. This same information might be presented with a different rhetorical function such as attributive, i.e. attributing certain information to guided missiles rather than using it to identify them.

The information in the propositions generally consists of objects and attributes from TEXT's database model, indicating attributes of the mentioned objects and sub-type relationships between the objects. Some of the rhetorical functions allow other sorts of information. Inference propositions, for example, can indicate comparisons between database values:

```
(inference OCEAN-ESCORT CRUISER
           (HULL_NO (1 2 DE) (1 2 CA))
           (smaller DISPLACEMENT)
           (smaller LENGTH)
           (PROPULSION STMTURGRD STMTURGRD))
```

Here TEXT infers that ocean escorts have smaller length and displacement than cruisers, that the two kinds of ships have the same form of propulsion and that their hull numbers differ in their first two letters.

The strategic component also produces focus information for each proposition to insure that the individual sentences will form a coherent paragraph when combined. Following Sidner's model [Sidner 83], TEXT indicates a discourse focus and potential focus list for each proposition. The tactical component uses this information to decide when to pronominalize and what sentence-level syntactic structure to use.

[1] No attempt was made to investigate changing the overall division into strategic and tactical components. In part this was because the task of adapting the MUMBLE system to work with an independently developed text planner seemed like an interesting experiment in itself. Also, TEXT's strategic component was in the process of being ported from a VAX to a Symbolics 3600, and was thus already in a state of flux.

2 Adapting MUMBLE to TEXT

MUMBLE is a general-purpose generation framework which has been used with several domains and message representations[McDonald 83b,Karlin 85].[2] MUMBLE-based systems are constructed out of three components: the interpreter, the grammar, and the dictionary. The interpreter controls the overall generation process, co-ordinating the propagation and enforcement of constraints and the (incremental) translation of the message.[3] The grammar enforces grammatical constraints and maintains local grammatical information. The dictionary indicates, for each term in the vocabulary of the message formalism, the various ways it can be expressed in English. In adapting MUMBLE to this new domain, each of these three components had to be modified to a different degree.

2.1 The Interpreter

The interpreter is a domain-independent embodiment of MUMBLE's approach to generation [McDonald 83a]. The translation process is guided by a depth-first traversal of the surface structure tree. Each position in the tree has one or more labels, which may indicate procedures to be run when the traversal enters or leaves that position. The leaves of the tree will either be words, which are printed out after morphological processing, or pieces of the original message. In the latter case, the interpreter looks up the message in the dictionary to find a realization for the message that satisfies the local constraints. The result is a new piece of surface structure tree which is spliced into the tree, possibly with part(s) of the original message as new leaves. In this way, the entire message is gradually translated and printed out.

Because the interpreter actually does some additional work beyond guiding the generation process, some modification to it was required. In particular, the routine that handles word morphology needed changes to the way it determined noun phrase plurality. A noun phrase was considered to be plural if it was derived from a message element that represented more than one object. This was adequate when the domain contained only specific objects, as has been the case in past uses of MUMBLE. In TEXT, however, many terms represent generic concepts, e.g. SHIP, which represents the concept of a ship rather than any particular ship. Generic concepts can be expressed using either singular or plural, for example "A ship is a water-going vehicle" vs. "Ships are water-going vehicles". Thus the morphology routine had to be modified to look at the surface structure tree to see how the term had actually been realized. (The grammar and dictionary also had to be modified to always explicitly mark plural noun phrases in the tree). This was the only modification necesary to the interpreter.

However, not all of the interpreter was used. In addition to the traversal and incremental expansion of the surface structure tree, MUMBLE provides a mechanism for subsequent messages to be combined with the original message as it is translated. This is done via

[2]The version of MUMBLE used with TEXT dates from March 1985 and was originally set up to translate the output of the GENARO scene description system[McDonald 83a].

[3]A "message" is simply an expression that the text planner (here TEXT's strategic component) sends to MUMBLE to be translated. This is the same as a "realization specification" in [McDonald 83a].

"attachment points"[McDonald 85] that are marked in the tree; a new message from the planner can be added at an attachment point if there is a way to realize it that satisfies the attachment point's grammatical constraints. For example, in translating messages from GENARO, MUMBLE puts an ATTACH-AS-ADJECTIVE attachment point before the head noun in noun phrases. This allows MUMBLE to combine the messages such as (introduce house_1) and (red house_1) and generate the single sentence "This is a picture of a red house" instead of "This is a picture of a house. It is red."

This attachment mechanism is not used with the TEXT output.[4] Originally this decision was made because TEXT's strategic component organizes its messages into sentence-size packets (the propositions), and there seemed little reason to split these up and then have MUMBLE recombine them.

It turned out, though, that there was one case where attachment points would have been useful. The attribute-value pair (TARGET-LOCATION X) (where X is the type of target location, e.g. SURFACE or WATER) can be translated as either "a target location <X as a prep. phrase>" or "a <X as an adjective> target location". The latter form is preferred, but can only be used if X can be realized as an adjective. Thus MUMBLE can produce "a surface target location", but must resort to "a target location in the water". The problem is that since the interpreter traverses the tree in depth-first order, MUMBLE must decide which form to use for (TARGET-LOCATION X) *before* determining whether X has a realization as an adjective. This is the one case where it was necessary to circumvent MUMBLE's control strategy. Attachment points could have solved this problem; the value X could have been a separate message which would have been attached ahead of "target location" only if it had a possible realization as an adjective.

Unfortunately, there was a problem that prevented the use of attachment points. Attachment can be constrained so that the result will be grammatical and so that the attached message will be together with the proper objects. For example, (red house_1) will only be attached as an adjective in a noun phrase describing house_1. But there was no principled way to force several messages to be combined into a single phrase. To see why this is a problem, consider a simple rhetorical proposition:

```
(identification SHIP WATER-VEHICLE (restrictive (TRAVEL-MODE SURFACE)))
```

("restrictive" indicates that this attribute distinguishes SHIP from other kinds of WATER-VEHICLE.) This is intended to produce something like "a ship is a water-going vehicle that travels on the surface". There are really two pieces of information here: that ships are water-going vehicles, and the ships travel on the surface. If we separate these out, the first would become (identification SHIP WATER-VEHICLE), and the second would become something like (attributive SHIP (TRAVEL-MODE SURFACE)). The problem is that there is no way to force MUMBLE to combine these back to get something like the original sentence. Instead, MUMBLE might translate these as "A ship is a water-going vehicle. Ships travel on the surface." The precise characterization of ships has been diluted. Even worse, if the next proposition is about ships, the travel-mode information may be

[4] Actually, attachment points are used to attach each proposition as a new sentence. This is simply a convenience to allow MUMBLE to be invoked once on a list of propositions; the results are exactly as they would be if MUMBLE were invoked individually on each proposition.

combined with it instead, completely destroying the rhetorical structure intended by the strategic component.

Of course, there is no immediately apparent advantage to splitting up identification propositions (although it does suggest the possibility of letting more of the structural decisions be made by MUMBLE). But the same problems arise in trying to solve the problem with `(TARGET-LOCATION X)` discussed above. Attachment would allow the system to choose correctly between "a surface target location" and "a target location on the surface". But then instead of "The missile has a surface target location. Its target location is indicated by the DB attribute DESCRIPTION", MUMBLE might produce "The missile has a target location. Its surface target location is indicated by the DB attribute DESCRIPTION."

What is needed is a way to constrain the attachment process to build several messages into a single phrase. In fact, this capacity *has* been added to MUMBLE, although it is not present in the version used with TEXT [McDonald, personal communication]. It is possible to create "bundles" of messages that can have additional constraints on their overall realization while allowing the individual messages to be reorganized by the attachment process. This facility would make it feasible to use attachment with TEXT.

2.2 The Grammar

A MUMBLE grammar is not simply a declarative specification of valid surface structure like the rules in a context-free grammar. Rather, it consists of procedures that enforce (local) constraints and update info about current grammatical environment. The grammar provides the low-level control on realization as the interpreter traverses the tree. Grammar in the more conventional sense is a by-product of this process.

The grammar operates via "constituent-structure labels". These labels are placed on positions in the surface structure tree to identify their grammatical function. Some, such as `adjective` and `np`, are purely syntactic. Others, such as `compound` and `name`, have more of a semantic flavor (as used with TEXT). Labels constrain the generation process through an associated "grammatical constraint". This is a LISP predicate that must be satisfied by a proposed realization. Whenever the interpreter tries to translate a message, it checks that the constraints associated with all the labels at the current tree position are satisfied. These constraints can depend on both the proposed realization and the current environment. The labels also provide for local operations such as propagation of constraints through the tree and production of purely grammatical words such as the "to" in infinitival complements and the "that" in relative clauses. As with the constraints, this is done by associating procedures with labels. Each label has several "grammar routines" to be run at various times (such as when the interpreter enters or leaves a node, or after a message is realized). For example, the `rel-clause` label prints "that" when the interpeter enters a node it labels.

The labels handle local aspects of the grammar; global aspects are managed via "grammar variables". These keep track of global information (i.e. information needed at more than one tree position). For example, there are grammar variables that record the current

subject and the current discourse focus. These "variables" are actually stacks so that embedded phrases can be handled properly. The grammar variables are maintained by the grammar routines associated with the labels. The `clause` label, for example, updates the current subject whenever the interpreter enters or leaves a clause. The grammar variables enable information to be passed from one part of the tree to another.

Adapting MUMBLE to TEXT required considerable modification and extension to the grammar. A number of new syntactic structures had to be added. Some, such as appositives, simply required adding a new label. Others were more complex; relative clauses, for example, required a procedure to properly update the current subject grammar variable (if the relative pronoun is serving as the subject of the relative clause) as well as a procedure to produce the initial "that". Also, some of the existing grammar had to be modified. Post-nominal modifiers, for example, previously were always introduced via attachment and realized as prepositional phrases. When working from TEXT, they are introduced as part of the original noun phrase, and they can sometimes be realized as relative clauses, so the constraints had to be completely redesigned.

The grammar was also augmented to handle some constraints that were more semantic than syntactic. These were included in the grammar because it is the only mechanism by which decisions made at one place in the tree can affect subsequent decisions elsewhere. In fact, there is really nothing inherently grammatical about the "grammar"; it is a general mechanism for enforcement of local constraints and propagation of information through the tree. It serves well as a mechanism for enforcing grammatical constraints, of course, but it is also useful for other purposes. For example, the grammar variable `current-entity-type` keeps track of whether the current clause is dealing with specific or generic concepts.

2.3 The Dictionary

The dictionary stores the various possible ways each kind of message can be realized in English. Dictionary entries provide the pieces of surface structure that are organized by the interpreter and constrained by the grammar. The dictionary has two parts: a look-up function and a set of "realization classes" (or "rclasses"). The look-up function determines which rclass to use for a message and how to construct its arguments (which are usually either sub-parts of the message or particular words to use in the English realization of the message). An rclass is a list of possible surface structures, generally parameterized by one or more arguments.

The look-up function is intended to be domain-dependent. However, the look-up function that was developed for GENARO, which has a fairly simple keyword strategy, seemed adequate for TEXT as well. The keyword is the first element of the message if the message is a list; otherwise it is the message itself. The function then simply looks up the keyword in a table of terms and rclasses. Using an existing function was convenient, but it did cause a few problems because it required that keywords be added to TEXT's formalism in a few cases. For example, numbers had to be changed to `(number #)` so they would have a keyword. Some straightforward modifications to the look-up function, however, would allow MUMBLE to generate from the original TEXT formalism.

The realization classes vary greatly in their generality. Some of them are very general. The rclass SVO, for example, produces simple transitive clauses; the subject, verb, and object are arguments to the rclass. At the other extreme, the rclass TRAVEL-MEANS-CLASS is only useful for a particular attribute as used by TEXT; even if another system had an attribute called TRAVEL-MEANS, it is unlikely to mean exactly the same thing.

Intuitively, it might seem that there would be a number of general realization classes like SVO. In fact, though, SVO was the *only* pre-existing rclass used in that was used for TEXT. None of the other rclasses proved useful.

One source of this lack of generality is that concepts that seem similar are often expressed quite differently in natural language. For example, of the eight generic attributes (e.g. TRAVEL-MEDIUM, TRAVEL-MEANS, and TARGET-LOCATION) in the dictionary, three require special rclasses because the general translation won't work for them. Inside TEXT's domain model, TRAVEL-MEDIUM and TRAVEL-MEANS are considered similar sorts of concepts. But in English, the two concepts are expressed differently. TEXT's notion of generic attribute simply doesn't correspond to any natural linguistic category.

Furthermore, different message formalisms will tend to capture different generalizations. GENARO can use a CONDENSE-ON-PROPERTY rclass[McDonald 83a] because it has a particular notion of what a property is and how it gets translated into English. TEXT doesn't have anything that exactly corresponds to GENARO's properties (and even if it did, it couldn't condense things because the properties would be buried inside the rhetorical propositions).

The crux of the matter is that while there are linguistic generalizations that might be captured in realization classes, they usually cut across the grain of the classes of expressions in a message formalism, and cut differently for different formalisms. Thus whatever generalizations can be encoded into the rclasses for one formalism are unlikely to be useful with a different formalism.

For example, TEXT can produce attribute expressions of the form:

```
(HULL_NO (1 2 DE))
```

which means, roughly, "characters 1 through 2 of the HULL_NO are DE". This is a very idiosyncratic sort of message; it is unlikely that another (independently developed) text planner would have a message form with even the same meaning, let alone the same syntax. Thus the dictionary entry for this message is unlikely to be of use with any system other than TEXT. Many of TEXT's messages were similarly idiosyncratic, because its message formalism was designed around the needs of its particular task. Similarly, other generation systems will have their own idiosyncratic message formalism. Thus they will need their own highly specific dictionaries to work with MUMBLE.

3 Using MUMBLE to produce text

3.1 Examples from TEXT

The new MUMBLE-based tactical component has been very successful. It can process all of the examples in the appendix to [McKeown 85] and produce comparable English text. Furthermore, it can process all 57 sentences in the appendix in about 5 minutes; the old tactical component took that long to produce a single sentence.

For example, TEXT's strategic component responds to a request to describe the ONR database with:

```
(attributive db OBJECT (name REMARKS))
(constituency OBJECT (VEHICLE DESTRUCTIVE-DEVICE))
(attributive db VEHICLE
              (based-dbs (SOME-TYPE-OF TRAVEL_MEANS)
                         (SOME-TYPE-OF SPEED_INDICES)))
(attributive db DESTRUCTIVE-DEVICE
              (based-dbs (SOME-TYPE-OF LETHAL_INDICES)))
```

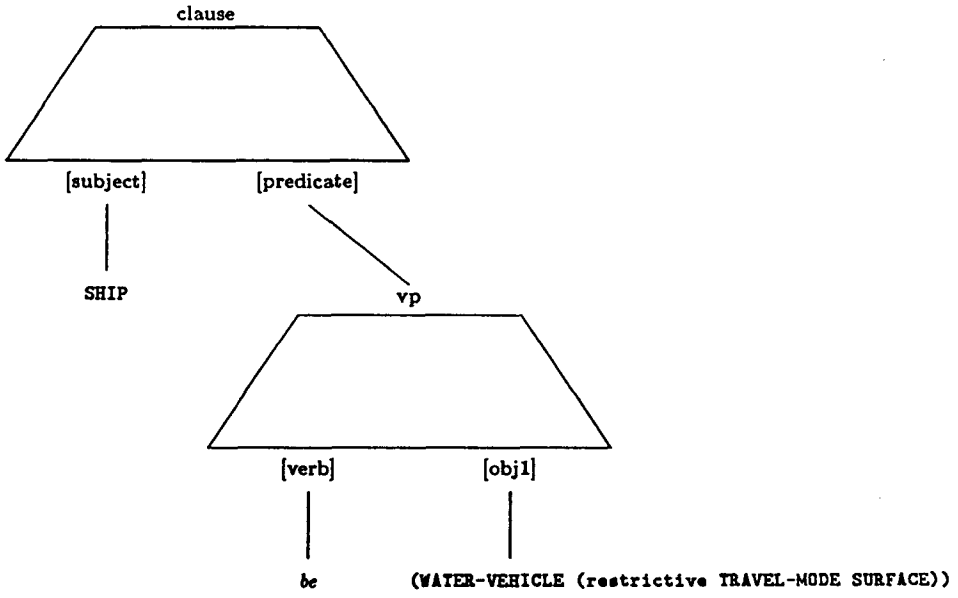
which is then translated into English by MUMBLE as:

> All entities in the ONR database have DB attributes REMARKS. There are 2 types of entities in the ONR database: vehicles and destructive devices. The vehicle has DB attributes that provide information on SPEED_INDICES and TRAVEL_MEANS. The destructive device has DB attributes that provide information on LETHAL_INDICES.

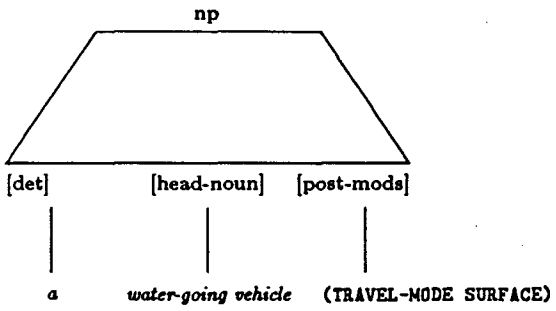
This translation is guided and controlled by the various sub-components that make up the MUMBLE tactical component, as can be seen in a more detailed example. The message:

```
(identification SHIP WATER-VEHICLE (restrictive TRAVEL-MODE SURFACE))
```

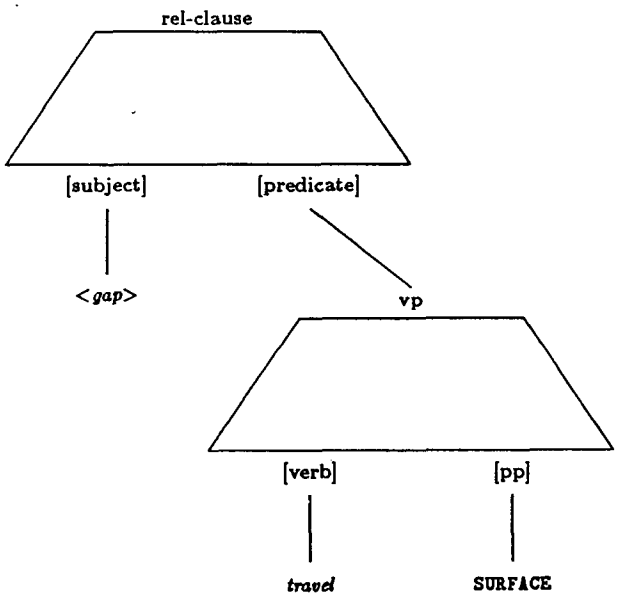
when received by MUMBLE, is first looked up in the dictionary, which indicates that the overall structure of the sentence will be:



The interpreter then traverses this (partially filled-out) surface structure tree, soon reaching the still untranslated message element SHIP. The first possibility listed for this in the dictionary is the noun phrase "a ship"; since no constraints rule it out, this choice is selected. The interpreter continues, printing the words "a" and "ship" as it reaches them. The morphology routine converts "be" to "is" by checking the number of the current subject and whether any deviation from simple present tense (the default) has been arranged for. Next the interpreter reaches the object, another message element which is translated (via dictionary lookup) as:



"A" and "water-going vehicle" are simply printed when passed through. The treatment of (TRAVEL-MODE SURFACE is more complicated. This message element can be translated in many ways, such as a noun phrase, a bare noun, a verb phrase, and so on. The post-mods label, however will allow only two possibilities: a prepositional phrase or a relative clause. Since the dictionary indicates that relative clauses are preferred over prepositional phrases (for this message) and there are no other constraints blocking it, the relative clause form is chosen:



The interpreter continues on through the relative clause in a similar fashion, eventually

producing "that travels on the surface". (Note, incidentally, that the word "that" is not explicitly in the tree; rather it is printed out by an attached routine associated with the `rel-clause` label.) The complete translation produced by MUMBLE is:

> A ship is a water-going vehicle that travels on the surface.

All three elements of the overall MUMBLE framework have worked together to produce the final English text.

3.2 Mumble and the Generation Process

The fundamental constraint that MUMBLE places on generation is, of course, that it is deterministic; this is the guiding principle driving its design, and has been discussed at length elsewhere[McDonald 83b,McDonald 83a]. There are, however, several other interesting constraints that MUMBLE places on the overall design of the generation process:

1. **The information used to guide the generation process is centered around the message formalism, not language.**
   MUMBLE's knowledge of how language expresses things is stored in the dictionary, organized around the possible expressions in the message formalism. Thus the "dictionary" does not list meanings of words, but rather possible (partial) phrases that can express a message. Similarly, the grammar is not set up primarily to express whether a sentence is grammatical but rather to constrain the choice of realizations as the sentence is generated. The grammatical constraints depend in part on the message being translated and the current grammatical environment (i.e. the grammar variables), none of which is preserved in the generated English sentence. Thus it may not be possible to tell whether a given sentence satisfies the grammar's constraints (at least without knowing a message it could have been generated from).

   This organization is a natural consequence of MUMBLE's purpose: to generate text. In language understanding, it is important to know about language, because that is what the system must be able to decipher. MUMBLE is also set up to know about its input, but its input is the message formalism, not natural language. What MUMBLE needs to know is not what a particular word or construction means, but rather when to generate it.

2. **Generation is incremental and top-down.**
   Large messages are partially translated incrementally, with sub-messages left to be translated later as the interpreter reaches them. Thus it is easy for large-scale structure to influence more local decisions, but harder (or impossible) for local structures to constrain the global structure that contains them. This asymetry is a direct consequence of determinism; *something* has to be decided first.

3. **Constraints can be associated both with the surface structure being built up and with possible realizations.**
   Thus the existing structure can constrain what further structures are built, and

candidate structures can constrain where they can be placed. This allows some of the bidirectionality that would seem to be ruled out by determinism. For example; transitive verbs can insist on only being used with direct objects, and verb phrases with direct objects can insist on getting transitive verbs. Note though that the decision to use a transitive verb phrase would still be made first, before the verb was selected.

4. **Constraints are largely local, with all global constraints anticipated in advance.**
   Most constraints are handled locally by constraint predicates that are attached to the surface structure tree or to the possible realization. Any global constraints must have been anticipated and prepared for, either by passing information down to the local node as the tree is traversed, or by storing the information in globally accessable grammar variables. Furthermore, all constraints are still locally enforced; global information can only constrain decisions if there are local constraints that use it.

4 Conclusion

The new MUMBLE-based tactical component has been very successful; it produces equivalent English text approximately 60 times faster than TEXT's old tactical component. Its construction, however, required modifications to each of the three parts of MUMBLE: the dictionary needed new entries for the new types of messages that TEXT produced; the grammar needed expansion to handle additional constructions and to implement new constraints that were needed for TEXT; and the interpreter was modified to handle a new criterion for noun phrase number. Furthermore, the new component sheds some light on how MUMBLE organizes the generation process and the consequences of its commitment to determinstic generation.

References

[Bossie 81] Bossie, Steve. *A Tactical Component for Text Generation: Sentence Generation Using a Functional Grammar.* Technical Report MS-CIS-81-5, CIS Department, University of Pennsylvania, Philadelphia, PA, 1981.

[Karlin 85] Karlin, Robin. *Romper Mumbles.* Technical Report MS-CIS-85-41, CIS Department, University of Pennsylvania, Philadelphia, PA, 1985.

[Kay 79] Kay, Martin. Functional Grammar. In *Proceedings of the 5th Annual Meeting of the Berkeley Linguistic Society.* 1979.

[McDonald 83a] McDonald, David D. Description Directed Control: Its Implications for Natural Language Generation. In Brady, N. (editor), *Computational Linguistics*, pages 111–129. Pergamon Press, 1983.

[McDonald 83b] McDonald, David D. Natural Language Generation as a Computational Problem. In Brady, M. and Berwick, Bob (editors), *Computational Models of Discourse*, pages 209–265. MIT Press, 1983.

[McDonald 85] McDonald, David D. and Pustejovsky, James. TAGs as a Grammatical Formalism for Generation. In *Proceedings of the 23rd Annual Meeting of the ACL*, Association for Computational Linguistics, pages 94–103. Chicago, 1985.

[McKeown 85] McKeown, Kathleen R. *TEXT GENERATION: Using Discourse Strategies and Focus Constraints to Generate Natural Language.* Cambridge University Press, 1985.

[Sidner 83] Sidner, C. L. Focusing in the Comprehension of Definite Anaphora. In Brady, M. and Berwick, Bob (editors), *Computational Models of Discourse*, pages 267–329. MIT Press, 1983.