David D. McDonald

This is a description of Mumble's approach to natural language generation, excerpted from a technical survey of generation entitled "Natural Language Generation: complexities and techniques," which will appear in Nirenburg (ed.) **Theoretical and Methodological Issues in Machine Translation**, Cambridge University Press, to appear 1986.

8. Multi-Level, Description Directed Generation

The principal deficit of the direct replacement approach is its difficulties with grammar, i.e. the awkwardness of maintaining an adequate representation of the grammatical context, or of carrying out grammatically mediated text-level actions such as producing the correct syntactic form for an embedded clause. In other respects, however, the message-directed control flow that drives direct replacement has a great deal to recommend it. Compared with grammar-directed control schemes, message-directed control is more efficient, since every action will contribute to the eventual production of the text. Message-directed control also gives a planner a very clear semantic basis for its communication to the realization component, since the message can be viewed simply as a set of instructions to accomplish specific goals. The question then becomes: is there a way of elaborating the basic, message-directed framework so as to overcome the deficits that plague direct replacement approaches while still keeping the computational properties that have made it attractive?

A number of generation researchers have independently choosen the same solution: to interpose a level of explicitly linguistic representation between the message and the words of the text (McDonald 1975, 1984; Kempen and Hoenkamp 1982; Jacobs 1985; Swartout 1984). They believe that employing a syntactic description of the text under construction is the most effective means of introducing grammatical information and constraints into the realization process, in particular, that it is a better locus for grammatical processing than a separately stated, active grammar.

The specifics of their individual treatments differ, but a common thread is clearly identifiable: Realization is organized as choices made by specialists, where the form of the choice--the output of the specialist--is a linguistic representation of what is to be said, i.e. a structural annotation of the syntactic relations that govern the words (and embedded conceptual elements) to be said, rather than just a list of words. These representations are phrase structures of one or another sort--hierarchies of nodes and constituents--of essentially the same kind that a theoretical linguist would use. They employ functional terms like "subject" and "focus", and are most aptly characterized as a kind of "surface structure" in the generative linguist's sense, e.g. they undergo no derivation, and are a proper and complete description of the syntactic properties of the text that is produced.

It will be convenient to restrict the present discussion to only one examplar of this approach; taking advantage of an author's prerogative, I will describe my own (c.f. McDonald 1984; McDonald & Pustejovsky 1985; McDonald, Pustejovsky & Vaughan 1986). As it is the historical outgrowth of a direct replacement system,[1] it will be useful to organize the discussion in terms of how it extends that approach and addresses its deficits. This will

[1] This author's interest in natural language generation began in 1971 while he was working on extentions to the grammar and parser in Winograd's SHRDLU program. As already discussed, SHRDLU employed a classic direct replacement technique for its generation. It was observations of the shortcomings of that design that were the original motivation for the research. The influences of systemic grammar and data-directed programming style also stem from that time.

be folded into the standard description of how it deals with the three general concerns one should have in examining a generation system: how it organizes its knowledge of grammar; what its control structure is; and what its approach to realization is.

Referring to our approach as "multi-level, description-directed generation" emphasizes specific features of its architecture and control protocols that we consider important; it is, however, too large a phrase to use conveniently. The name of the computer program that implements the design, MUMBLE (McDonald 1977, 1983), will serve as a compact, agentive reference. Characterizing MUMBLE as multi-level draws attention to the fact that it carries out operations over three explicitly represented levels of representation simultaneously: message, surface structure, and word stream. Description-directed is the name we have given to its control protocol, which is a specialization of the common programming technique known as data-directed control. Under this protocol, the data in the representations at the three levels is interpreted directly as instructions to the virtual machine that constitutes the generator proper. Since each of these representational structures is also a valid description of the text at its own level of abstraction and theoretical vocabulary, this characterization of the protocol emphasizes the fact that the particulars of how the person developing messages or syntactic structures chooses to design them has immediate consequences for the generator's performance (McDonald 1984). The feedback that this gives a developer has proven to be invaluable in refining the notations and their computational interpretations in all parts of the system.

MUMBLE's virtual machine is the embodyment of our computational theory of generation. It consists of three interleaved processes that manage and carry out the transitions between the representational layers. (1) Phrase structure execution interprets the surface structure, maintaining an environment that defines the grammatical constraints active at any moment, and producing the word stream as its incremental output. (2) Attachment interprets the message, transferring its component units to positions within the surface structure according to the functional relationships between them and their role in the message. (3) Realization takes the individual elements of the message into surface structure phrases by selecting from linguistically motivated classes of parameterized alternative forms. A minor fourth process, operating over the word stream, morphologically specalizes individual words to suit their syntactic and orthographic contexts (e.g. the article"a" going to "an" before vowels); later versions of MUMBLE that produce speech should be much more active at this level.

Thus, as seen by the developer of a text planner that would pass messages to MUMBLE for it to produce texts from, the virtual machine appears as a very high level, task-specific language, with its own operators and intermediate representations. To a lesser extent this is true also for the linguist writing generation-oriented grammars for MUMBLE to execute, since the virtual machine includes no presumptions as to what specific syntactic categories, functional relations, or syntactic constructions the natural language includes. Instead it supplies a notation for defining them in terms of primitive notions including the dominates and proceeds relations of phrase structure, bound thematic relations, configural regularities such as head or complement from X-bar theory; and the tree combination rules of Tree Adjoining Grammars (Kroch & Joshi, 1985).

As a message-directed design, MUMBLE is best discussed by reference to a concrete example message, situation, and resulting output text. To miminize the distraction that introducing an actual underlying program from one of our generation projects would entail, a relatively obvious excerpt from a message will have to suffice. The figure shows a generated output paragraph describing a legal case from the UMass Counselor Project (McDonald & Pustejovsky 1986). The structure below it is the message responsible for its second sentence, which details the events that were relevant to the court's decision. Using this example, we will look at MUMBLE's knowledge of grammar: how it is manifest, and how it has its effects, interleaving discussion of realization and control at convenient places.

"In the Telex case, Telex was sued by IBM for misappropriating trade secrets about its product Merlin. One of the managers of the Merlin development project, Clemens, left IBM to work for Telex, where he helped to develop Telex's competing product, the 6830. The key fact in the case was that Clemens brought a copy of the source code with him when he switched jobs. The court held for IBM."

```
(temporal-sequence
  (left-to-work-for (*‹role *‹project-manager Merlin›› *‹Clemens›)
                    (named-company *‹IBM›)
                    (named-company *‹Telex›))
  (helped-to-develop (named-person *‹Clements›)
                     (*‹kind product› *‹competition-by *‹Telex››
                                      *‹name "6830"›) ))
```

As previously discussed, one of the concomitant features of a message-directed approach is that items[2] directly from the underlying program are part of the messages. (These are indicated here by enclosing angle brackets, *‹...›.) Once in a message, such items become instructions to the generator, and as such need interpretations, i.e. associated functions from the item, and the linguistic and pragmatic environment, to the surface specification of some text or text fragment. However, considered in terms of the space of texts that might realize them, real program objects are large and vague as present day programmers tend to use them: they stand in many different relationships to other objects and to the underlying program's state, and consequently can have many different interpretations depending on the context and the speaker's intent.

We take it to be part of the job of a text planner to choose among these relationships and to indicate in the message the perspective from which an object is to be viewed. (The perspective on the first occurance of Clemens, for example, is indicated to be his role as (former) manager of the Merlin project.) Adopting a specific perspective often amounts to selecting a specific wording (often just of the lexical head, e.g. "manager"; but also entire conventional phrases such as "leave ‹employer1› to work for ‹employer2›"). These examples indicate that many of the terms in a message are surface lexical relations (e.g. "helped to develop") rather than a more abstract conceptual vocabulary; this has the deliberate corollary that syntactic realization will usually occur after key words have been chosen. The text planner must therefore understand a good deal about how alternative word choices cover the semantic fields of the situation it is trying to communicate, and what emphasis and what presupposed inferencing by the audience a given choice of wording will convey. This appears to us to be a choice that is best made at a conceptual level (i.e. during message construction), since it does not depend in any crutial way on the details of the grammatical environment, the arguments of Danlos (1984) notwithstanding (cf. McDonald et al. 1986).

Even though the key lexical choices for an item will have occurred before it has been syntactically realized, these message-level lexical decisions can draw on the grammatical context in which the text for it is going to occur. In particular, grammatical constraints imposed by the syntactic relations in which the text will stand will filter out grammatically

[2] The word "item", and at other times the word "object", is intended as a general term that denotes representational data structures in an underlying program without regard to the kind of real world entity that they model: individuals, kinds, relations, constraints, attributes, states, actions, events, etc.

inconsistent possibilities from the planner's choice set.[3] This is possible because the realization of messages is hierarchical, following the message's compositional structure top down, i.e. the message is interpreted much as a conventional program would be. The surface syntactic realization of the higher, dominating conceptual elements of the message is thus available to define and constrain the interpretations (i.e. linguistic realizations) of the lower, more embedded elements. This protocol for "evaluation" of arguments is known as normal order, and is in direct contrast with the previously discussed applicative order protocol used in most direct replacement designs.

The perspective that the text planner chooses to impose on an item from the underlying program is represented at the message-level by designating the realization class to be used for it. Realization classes are MUMBLE's equivalent of the "specialist programs" in direct replacement. They are linguistic entities rather than conceptual, and are developed by the designer of the grammar using control and data structures defined in the virtual machine. New underlying programs are interfaced to MUMBLE by developing a (possibly very minimal) text planner and assigning program items (or item types) to pre-defined realization classes. A relatively self-contained example of a class, "locative-relation", developed originally for use with Jeff Conklin's program for describing pictures of house scenes (see Conklin, 1984) is shown below:

```
(define-realization-class LOCATIVE-RELATION
  :parameters (relation arg1 arg2)
  :choices
    ( (Arg1-is-Relation-Arg2)
      "The driveway is next to the house"
      clause focus(arg1) )
    ( (Arg2-has-Arg1-Relation-Arg2)
      "The house has a driveway in front of it"
      clause focus(arg2) )
    ( (There-is-a-Arg1-Relation-Arg2)
      "There is a driveway next to the house"
      root-clause shifts-focus-to(arg1) )
    ( (Relation-Arg2-is-Arg1)
      "Next to the house is a driveway"
      root-clause shifts-focus-to(arg1)
      final-position(arg1) )
    ( (with-Arg1-Relation-Arg2)
      "...with a driveway next to it"
      prepp modifier-to(arg1) ))
```

[3] This filtering is automatic if the relevant parts of the text planner are implemented using the same abstract control device as MUMBLE uses for its own decisions, i.e. parameterized, pre-computed annotated choice sets of the sort employed for realization classes (see text). The descriptions of the lingustic character and potential of the choices that the annotation provides are the basis for filtering out incompatible choices on grammatical grounds, just as occurs at the syntactic level in selections within a realization class.

This technique is proving convenient in our own work with some simple text planners; however we can see a point where the requirement that the full set of alternatives be pre-computed may be unnecessarily limiting or possibly psychologically unrealistic, in which case an alternative design, presumably involving dynamic construction of the choices, will be needed and an alternative means of imposing the grammatical constraints will have to be found. For a discussion of another planning-level control paradigm that has been used with Mumble, see Conklin (1984) or McDonald & Conklin (1983).

The choices grouped together in a realization class will all be effective in communicating the conceptual item assigned to the class, but each will be appropriate for a different context. This context-sensitivity is indicated in the annotation accompanying the choice, for example "focus", which will dictate the grammatical cases and surface order given to the arguments, or the functional role "modifier-to", which will lead to realization as a postnominal prepositional phrase. These annotating characteristics indicate the contexts in which a choice can be used. They act both as passive descriptions of the choice that are examined by other routines, and as active test predicates that sample and define the pragmatic situation in the text planner or underlying program. Such terms are the basis of MUMBLE's model of language use--the effects that can be achieved by using a particular linguistic form; as such they play the same kind of role as the "choosers" or the controlling functional features in a systemic grammar like Mann's NIGEL.

The surface structure level, the source of grammatical constraints on realization, is assembled top down as the consequence of the interpretation and realization of the items in the message. In the example message (repeated below), the topmost item is a "sequence" of two steps, each of which is a lexicalized relation over several program objects on which a particular perspective has been imposed.

```
(temporal-sequence
  (left-to-work-for (*‹role *‹project-manager Merlin›› *‹name "Clemens"›)
                    (named-company *‹IBM›)
                    (named-company *‹Telex›))
  (helped-to-develop (named-person *‹Clements›)
                     (*‹kind product› *‹competition-by *‹Telex››
                                      *‹name "6830"›) ))
```

One of the goals of a multi-level approach is to distribute the text construction effort and knowledge throughout the system so that no level is forced to do more of the work than it has the natural capacity for. Thus for example in the interpretation of the first item the message, temporal sequence, MUMBLE is careful to avoid taking steps that would exceed the intent of the planner's instruction by being overly specific linguistically: As a message-level instruction, temporal-sequence says nothing about whether the items it dominates should appear as two sentences or one; it says simply that they occured after one another in time and that their realizations should indicate this. Since there is no special emphasis marked, this can be done by having them appear in the text in the order that they have in the message. The decision about their sentential texture is postponed until a linguistic context is available and the decision can be made on an informed basis.

This delay is achieved by having the Attachment process, which moves items from the message to the surface structure according to their functional roles, wait to position the second item of the sequence until the first has been realized. Only the first item will be moved into the surface structure initially, and it will appear as the contents of the second sentence as shown below. Note that a message item is not realized until it has a position, and then not until all of the items above it and to its left have been realized and the item has been reached by the Phrase Structure Execution process that is traversing the surface structure tree and coordinating all of these activities. By enforcing this discipline one is sure that all the grammatical constraints that could affect an item's realization will have been determined before the realization occurs, and consequently the virtual machine does not need to make provisions for changing an item's realization after it is finished (see figure one).

Considered as a function, a realization class such as "Left-to-work-for" specifies the surface form of a grammatically coherent text fragment, which is instantiated when the class is executed and a specific version of that phrase selected. Given its lexical specificity, such a class is obviously not primitive. It is derived by sucessive specializations of two, linguistically primitive subcategorization frames: one built around the verb class that

includes "leave" (shown below) and the other around the class containing "work for". The specialization is done by a definition-time currying operation wherein arguments to the subcategorization frames are bound to constants (e.g. the verb "leave"), producing new realization classes of reduced arity. On its face, a class built around variants on the phrase "‹employee› leaves ‹company1› to work for ‹company2›" is more appropriate to a semantic grammar (cf. Burton & Brown 1977) than to a conventional syntactic phrase structure grammar. This choice of linguistic modularity does however reflect the actual conceptual modularity of the underlying program that drives the example,[4] and we believe this is an important benefit methodologically.

```
(define-phrase subject-verb-locative (subj vb loc)
  :specification (clause
                   subject subj
                   predicate (vp
                               verb vb
                               locative-complement loc )) )
```

Comparing MUMBLE's organization of grammatical knowledge with that of the two grammar-directed approaches that have been discussed, we see that it resembles an ATN somewhat and a NIGEL-style systemic grammar hardly at all. ATN designs are based on procedurally encoded surface structures, which are executed directly; MUMBLE represents surface structure explicitly and has it interpreted. ATNs select the surface form to be used via a recursive, phrase by phrase, topdown and left to right consideration of the total set of forms the grammar makes available (i.e. alternative arc sequences), and queries the state of the underlying program to see which form is most appropriate. MUMBLE also preceeds recursively, topdown and left to right, but the recursion is on the structure of an explicitly represented message. Conceptual items or item types, through the the realization classes that the planner associates with them, control the selection and instantiation of the appropriate surface forms directly.

MUMBLE "packages" linguistic relations into constituent phrases; it does not provide an unbundled, feature-based representation of them as a systemic grammar does. It cannot, for example, reason about tense or thematic focus apart from a surface structure configuration that exhibits them. This design choice is deliberate, and reflects what we take to be a strong hypothesis about the character of linguistic knowledge. This hypothesis is roughly that the space of valid feature configurations (to use systemic terms) is smaller, less arbitrary, and more structured than a feature-heap notation can express (see McDonald et al. 1986 for details). Since our notation for surface structure incorporates functional annotations as well as categorical, and especially since it is only one of three representational levels operated over in coordination, we believe that organizing linguistic reasoning in terms of packaged, natural sets of relations will provide a great deal of leverage in research on text planning and computational theories of language use and communicative intention.

Nowhere in MUMBLE is there a distinct grammar in the sense of a set of rules for deriving linguistic forms from primitive features. Rather it manipulates a collection of

---

[4] As it happens, Leave-to-work-at is a primitive conceptual relation in the legal reasoning system that serves here as the underlying program (Rissland & Ashley, submitted). The causal model that the phrase evokes in a person, i.e. that working for the new company is the reason why the employee is leaving (cf. "John washed his car to impress his girlfriend") is encapsulated in this relation, and suppresses the causal model from consideration by the legal reasoner's rules. This encapsulation is deliberate. Reasoning systems should function at the conceptual level best suited to the task. This does however imply that some component of the natural language interface must now bridge the conceptual ground between the internal model and the lexical options of the language; see Pustejovsky (this volume) for a discussion of how this may be done.

predefined linguistic objects--the minimal surface phrases of the language and the composite phrases derived from them. The phrases are grouped into the realization classes, the projected linguistic images of different conceptual types and perspectives. When selected and instantiated to form the surface structure they take on an active role (through interpretation by the three processes), defining the order of further actions by the generator, defining the contraints on the realization of the embedded items from the message now at some of its leaf positions, and defining the points where it may be extended through further attachments from the message level. The figure below shows a snapshot of the surface structure for the first part of the text in the example, and can illustrate these points. At the moment of this snapshot, the Phrase Structure Execution process has traversed the structure up to the item *‹telex› and produced the text shown; its next action will be to have that item realized, whereupon the realizing phrase (an NP like the one for *‹IBM›) will replace *‹telex› in the surface structure and the process will traverse it and move on (see figure two).

The first thing to consider is the differences in the details of this surface structure representation compared with the more conventional trees used by generative grammarians. Two of these are significant in this discussion. The first is the presence of functional annotations over each of the constituents (indicated by labels inside square brackets). Terms like "subject" or "prep-complement" are used principally to summarize the grammatical relations that the constituents are in by warrant of their configurational positions, which makes these labels the source of most of the grammatical constraints on message item realizations. The functional annotations also play a role in the dynamic production of the word stream: Here this includes providing access to the subject when the morphological process needs to determine the person/number agreement for tensed verbs, and supplying grammatical function words like "of" or the infinitive marker "to" directly into the word stream.[5]

Formally the representation is not a tree but a sequential stream (as indicated by the arrows): a stream of annotated positions that are interpreted, in order, as instructions to the Phrase Structure Execution process. The grammar writer defines the interpretation an annotating label is to have, e.g. specifying control of morphological effects or function words, constraints to be imposed on realizations, or establishing salient reference positions (like the subject). Various useful technical details are expedited by defining the surface structure as a stream rather than a tree (see McDonald & Pustejovsky 1985b). The stream design provides a clean technical basis for the work of the Attachment process, which extends the surface structure through the addition of successive items from the message. The extensions are integrated into the active grammatical environment by breaking interposition links in the stream and kniting in the new items along with any additional covering syntactic nodes or functional constituent positions needed to correctly characterize the linguistic relationship of the new material to the old.

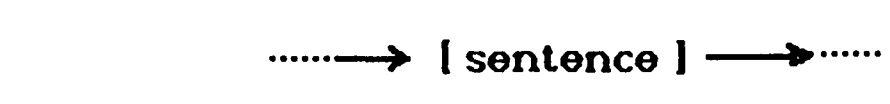
In the present example, the second item of the message's temporal sequence item, the lexicalized relation "helped-to-develop", remains unattached--its position in the surface

---

[5] Introducing the closed class words that indicate syntactic function into the text as an active consequence of traversing the corresponding part of the surface structure tree, rather than having them first appear in constituent positions at the tree's leaves, is an experimentally motivated design decision. It is intended to explore the consequences of employing computational grammars that distinquish the sources of closed and open class words: positing that the open class words have a conceptual source and the closed class "function" words a purely syntactic source. The two word classes are distinguished psycholinguistically, e.g. they have very different behaviors in exchange errors (see Garrett 1975); if this empirical difference can be given a successful computational account, then that account can serve to anchor other aspects of the grammar's design and eventually lead to psycholinguistic predictions derived from the consequences of the computational design (McDonald 1984).

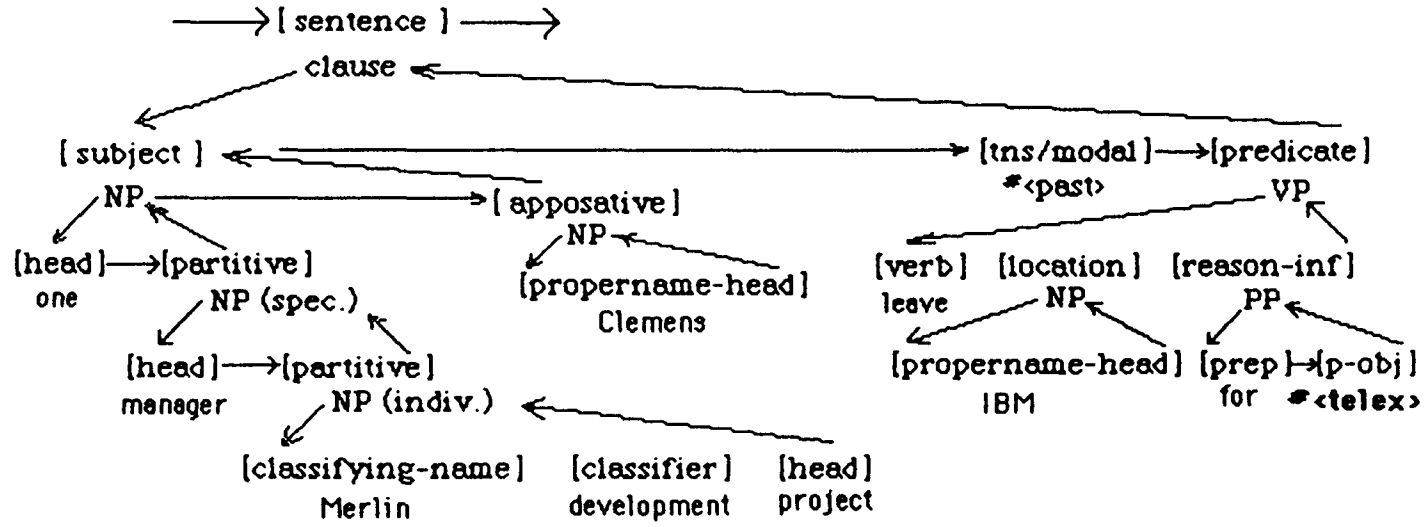
structure unestablished--until enough linguistic context has been established that a reasonable decision can be made about stylistic matters, e.g. whether the item should appear as an extension of the first item's sentence or start its own. Since the functional constraints on a temporal sequence's realization prohibit embedding the second item anywhere within the first, the only legal "attachment points" for it (i.e. links it could be knit in at) are on the trailing edge of the first item's sentence or as a following sentence. In terms of our theory of generation, attachment points are grammatical properties of phrasal configurations: places where the existing surface structure may be extended by splicing in "auxiliary" phrases (i.e. realizations of message items), for example adding an initial adjunct phrase to a clause or embedding the NP headed by "manager" inside the selector "one of". Every phrasal pattern (as indicated by the annotating labels) has specific places where it can be extended and still be a grammatically valid surface structure; the grammatical theory of such extensions is developed in studies of Tree Adjoining Grammars (Kroch & Joshi 1985).

What attachment points exist is a matter determined by the grammatical facts of the language; which points are actually used in a given situation is a matter of stylistic convention (see McDonald & Pustejovsky 1985a). In this case there is a very natural, compactly realized relationship between the first and second events: the final item in the realization of the first event, the Telex company, happens to be where the second event occurred. As neither clause is particularly complex syntactically, the attachment point that extends the final NP of the first event with a relative clause is taken and the second event knit into the surface structure there, to be realized when that position is reached in the stream.



The first item of the message in a top level position of the surface structure annotated as a 'sentence'

FIGURE ONE



Said so far:

"... One of the managers of the Merlin development project, Clemens left IBM for //"

FIGURE TWO