American Journal of Computational Linguistics *Microfiche 33 : 4*

A Framework for Writing Generation Grammars for Interactive Computer Programs

DAVID McDONALD

Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

ABSTRACT

This paper outlines the stucture and operation of the linguistic component from a language generation system in an interactive program. The component receives messages describing what is to be said formulated in the representation of the main program and produces fluent English utterances appropriate to the current discourse situation. The component is data-directed and uses a procedural grammar, organized as a set of strategies.

Interactive, specialist programs presently under development will need to produce fluent, intentional English utterances in responce to particular, complex situations. This creates a requirement for language generating facilities that is not faced in transformational grammar, mechanical translation programs, or paraphrase generating programs. As a component of an interactive, specialist program, the production of the English must be driven <u>directly</u> by the communicative intentions of the program and by the discourse situation.

We can imagine that the overall program consists of a number of cooperating modules - for parsing and interpreting what is said to it, for solving problems in its domain, for managing its memory, and, in particular, for generating utterances to communicate with its users. This generation component can be profitably viewed as having three aspects or "sub-components".

1) Situation/domain specialists that are activated when the program recognizes what situation it is in. They then decide what message will be produced. They will decide what effect on the listener is desired, and exactly what objects and relations are to be mentioned.

   For example, an appointment scheduling program might be told to "schedule a group meeting for Friday" and then find that a critical member of the group is unavailable. The situation specialists in the scheduling program are the ones to decide whether it is more appropriate to simply say "I can't", or whether to volunter information - "I can't; Mitch won't be back until Monday".

2) Models of the audience and the discourse situation to use in constructing utterances. There must be a record of the past conversation to guide in the selection of pronouns. Also, the program must have models of, and heuristics about what the audience already knows and therefore doesn't have to be told. This information may be very specific and domain dependent. For example, in chess, one can say "the white queen could take a knight". There is no need to say "a black knight", because this information is supplied by inferences from what one knows about chess - inferences that the speaker assumes the listener shares.

3) Linguistic knowledge about how to construct understandable utterances in the English language. Obviously, this information will include a lexicon associating objects and relations from the main program with strategies for realizing them in English (particular words, phrases,

syntactic constructions, etc.). There is also a tremendous amount of information which describes the characteristics of the English language and the conditions of its use. It specifies the allowable arrangements of strategies and what modifications or alternatives to them may be appropriate in particular circumstances.

Of the three aspects just described, my work has concentrated on the third. What follows is drawn from my thesis (McDonald '75) and from ongoing research.

The Linguistic Component

The linguistic knowledge required for generating utterances is put into one component whose job is to take a message from the situation specialists and construct a translation of that message in English. The messages are in the representation used by the main program and the situation specialists. The translation is done by a data-directed process wherein the elements and structure of the message itself provide the control.

The design of the linguistics component was arrived at independent of any particular main program, for the simple reason that no programs of adequate complexity were available at the time. However, at the present time a grammar and lexicon is being developed to use with at least two programs being developed by other people at MIT. They are an appointment scheduling program (Goldstein '75) and an advisor to aid users of MACSYMA (Genesereth '75). The short dialog below is an example of the degree of fluency we are hoping to eventually achieve. The dialog is between a scheduling program acting as an appointment secretary (P), and a student (S).

(S) I want to see Professor Winston sometime in the next few days.
(P) He's pretty busy all week. Can it wait?
(S) No, it can't. All I need is his signature on a form.
(P) Well, maybe he can squeeze you in tommorrow morning. Give me your name and check back in an hour.

Messages

Using the current message format and ignoring the details of the scheduler's representation, the phrase "maybe he can squeeze you in tommorrow" could have come from a message like this one, put together by one of the situation specialists.

```
        Message-1    features= ( prediction )
              event    (event actor <Winston>
                              action <fit person-into full schedule>
                              time <31-10-75,9am-12am>)
              hedge    <is possible>
              aim-at-audience  hedge
```

Messages have features describing the program's communicative intentions - what sort of utterance is this to be; what effect is it to have. Messages list the objects to be described (the right hand column) along with annotations for each object (left hand column) to show how they relate to the rest of the message. The phrases on the right in angle brackets represent actual structures from the scheduler with those meanings.

The Lexicon

Translation from the internal representaiton of a computer program to natural language has the same sort of problems as translating between two natural languages. The same concepts may not be available as primitives in both representations, and the conventions of the target language may require additional information that was not in the source. Generally speaking translation cannot be one for one.

What English phrase is best for a particular element in a program's message will depend on what is in the rest of the message and of what the external context is. In such circumstances, translation by table-lookup is inadequate. In this component, in order to allow all factors to be considered, the translation of each element is done by individualized procedures called "composers".

For each main program that the linguistic component becomes associated with, a lexicon must be created which will list the elements of the main program's representation that could appear in a message (i.e. "prediction","event","<Winston>", etc.). With each element is recorded the composer that will be run when the time comes to produce an English description for it (examples will be given shortly). Some composers may be applicable for a whole class of elements, such as "events". They would know the structure that all events have in common (e.g. actor, action, time) and would know how to interpret the idiosyncratic details of each event by using data in the lexicon associated with them.

The Grammar - strategies

The bulk of the grammar consists of "strategies". Strategies are associated with particular languages rather than with particular main programs as composers are. A given strategy may be used for several different purposes. A typical case is the strategy use-simple-present-tense: a clause in the simple present ("prices rise") may be understood as future, conditional, or timeless, according to what other phrases are present.

Each composer may know of several strategies, or combinations of strategies which it could use in describing an element from the message. It will choose between them according to the context - usually details of the element or syntactic constraints imposed by previously selected strategies. The strategies themselves do no reasoning; they are implemented as functions which the composers call to do all the actual construction of the utterance.

The Translation Process

At this point, the outline of the data-driven translation process can be summarized. A message is given for translation. The elements of the message are associated in a lexicon with procedures to describe them. The procedures are run; they call grammatical strategies; and the strategies construct the English utterance.

Of course, if this were all there was to it, the process would never run, because all of the subprocesses must be throughly coordinated if they are not to "trip over their own feet", or, for that matter, if ordinary human beings are to be able to design them. In a system where the knowledge of what to do is distributed over a large number of separate procedures, control structure assumes central importance.

Plans

Before describing the control structure, I must lay out some additional aspects of the design of the linguistics component. Messages are translated directly into English surface structure form. There is no interlingua or intermediate level of structure comparable to the deep structures of Transformational Grammar, or the semantic nets of Simmons (73) or Goldman (74).

Determining the appropriate surface structure, however, requires planning, if for no other reason than that the message can only be examined one piece at a time. The entire utterance must be organized before a detailed analysis and translation can get underway. As this is done, the "proto-utterance" is represented in terms of a sort of scaffolding - a representation of the ultimate surface structure tree insofar as its details are known with extensive annotation, explicit and implicit, to point out where elements that are not yet described may be positioned, and to implement the grammatical restrictions on possible future details as dictated by what has already been done.

The scaffolding that is constructed in the translation of each message is called its "plan". Plans are made up of syntactic nodes of the usual sort - clauses, noun groups, etc. - and nodes may have features in the manner of systemic grammar (Winograd '72). Nodes have subplans consisting of a list of named slots marking the possible positions for sub-constituents, given in the order of the eventual surface structure. Possible slots would be "subject", "main verb", "noun head", "pre-verb-adverb", and so on. The syntactic node types will each have a number of possible plans, corresponding to the different possible arrangements or sub-constituents that may occur with the different combinations of features that the node may have. Depending on the stage of the translation process, a slot may be "filled" with a pointer to an internal object from the message, a syntactic node, a word or idiom, or nothing.

The translation process

The translation is done in two phases. The second phase does not begin until the first is completely finished. During the first phase, a plan is selected and the elements of the message are transferred, largely untouched; to the slots of the plan and features added to its nodes. During the second phase, the plan is "walked" topdown and from left to right. Composers for message elements in the plan's slots are activated to produce English descriptions for the elements as they are reached in turn. Both processes are data-directed, the first by the particular contents of the message and the second by the structure of the plan and the contents of its slots.

There are sound linguistic reasons for this two stage processing. Most parts of a message may be translated in terms of very modular syntactic and lexical units. But other parts are translated in terms of relations <u>between</u> such units, expressed usually by ordering or clause-level syntactic mechanisms. The exact form of the smaller units cannot be determined until their larger scale relations have been fixed. Accordingly, the objective of the first phase is to determine what global relationships are required and to choose the plan, features, and positions of message elements within the plan's slots that will realize those relationships. Once this has been done, English descriptions for the elements can be made independent of each other and will not need to be changed after they are initially created.

One of the most important features of natural language is the ability to omit, pronominalize, or otherwise abbreviate elements in certain contexts. The only known rules and hUristics for using this feature are phrased in terms of surface structure configurations and temporal ordering. Because the second phase works directly in these terms, stating and using the available heuristics becomes a straightforward, tractable problem.

"Maybe he can squeeze you in tommorow morning"

The rest of this paper will try to put some flesh on your picture of how this linguistics component works by following the translation of the message given in the beginning to the sentence above. The message was this.

```
Message-1     features= ( prediction )
     event     (event actor <Winston>
                      action <fit person into full schedule>
                      time <31-10-75,9am-12am>)
     hedge     <is possible>
     aim-at-audience  hedge
```

The intentional features of a message tend to require the most global representation in the final utterance, because that is where indicators for questions, special emphasis, special formats (e.g. comparison), and the like will be found. By convention then, the composers associated with the intentions are given the job of arranging for the disposition of all of the message elements. The total operation of phase one consists of executing the composer associated with each feature, one after the other.

This message has only one feature, so its composer will assume all the work. The linguistics component is implemented in MACLISP, features (and annotations and slots and nodes) are atoms, and composers are functions on their property lists.

```
Prediction
     composer-with  (lambda ... )
```

Making a prediction is a speech act, and we may expect there to be particular forms in a language for expressing them, for example, the use of the explicit "will" for the future tense. Knowledge of these would be part of the composer. Inside the main program, or the situation specialist, the concept of a prediction may always include certain parts: what is predicted, the time, any hedges, and so on. These part are directly reflected in the makeup of the elements present in the message, and their annotations mark what internal roles they have. There does not need to be a direct correspondence between these and the parts in the linguistic forms used, the actual correspondence is part of the knowledge of the prediction composer.

Typically, for any feature, one particular annotated element will be of greatest importance in seting the character of the whole utterance. For predictions, this is the "event". The prediction composer chooses a plan for the utterance to fit the requirements of the event-element. The realization of any other elements will be restricted to be compatible with it.

The prediction composer does not need to know the element's linguistic correlates itself, it can delegate the work to the composer for the element itself. The element look like this.

```
(event actor <Winston>
       action <fit person into full schedule>
       time <31-10-75,9am-12am>)
```

The first word points to the name of the composer, and the pairs give particular details. There is nothing special about the words used here (actor, action, time), just as long as the composer is designed to expect the information in those places that the message-assembler wants to put it. The event composer's strategy is to use a clause, and the choice of plan is determined by the character of the event's "action".

The action is "<fit person into full schedule>", and it will have two relevant properties in the lexicon: "plan", and "mapping". Plan is either the name of a standard plan to be used, or an actual plan, partially filled with words (i.e. it can be a phrase). "Mapping" is an association list showing how the subelements of the message are to be transferred to the plan.

```
<fit person into full schedule>
   PLAN
        node-i   (clause transl particle)
         slots frontings nil
               subject nil
               vg  node-j  (verb-group particle)
                slots modal  nil
                      pre-vb-adv  nil
                      mvb  "squeeze"
                      prt  "in"
               object1  <person being talked about>
               post-modifiers  nil
   MAPPING
        (( actor  subject )
         ( time post-modifiers))
```

The event composer proceeds to instanticte the nodes in the phrase and make the transfers; the prediction composer then takes the resulting plan, and makes it the plan of the whole utterance.

Two message elements remain, but actually there is only one, because "aim-at-audience" is supplying additional information about the hedge. The annotation means that the contents of the hedge (<is possible>) are more something that we want to tell the audience than a detail of the prediction. This will affect how the element is positioned in the plan.

The prediction composer looks in the lexicon to see what grammatical unit will be used to realize <is possible>, and sees, let us say, two possibilities involving different configurations of the adverb "maybe" and the modal "can be able to", with the differences hinging on the placement of the adverb. Theoretically, adverbs can be positioned in a number of places in a clause, depending on their characteristics. In this instance, the choice is forced because of a heuristic written into the grammar of adverbs and accessible to the composer, that says that when the intent of an adverb is directed to the audience, it should be in the first position (the "frontings" slot). This choice implies putting "can" in the modal slot directly. The alternative with "maybe" in the pre-vb-adv slot would have necessitated a different form of the modal, yielding "may be able to". These details would have been taken care of by syntactic routines associated with the verb group node.

All the message elements have been placed and the first phase is over. The plan is now as below.

```
node-1   (clause transl particle)
  slots frontings  "maybe"
        subject  <winston>
        vg  node-2  (verb-group particle)
          slots modal  "can"
                pre-vb-adv  nil
                mvb  "squeeze"
                prt  "in"
        object1  <person being talked about>
        post-modifiers  nil
```

The second phase controller is a simple dispaching function that moves from slot to slot. "Frontings" contains a word, so the word is printed directly (there is a trap for morphological adjustments when necessary). "Subject" contains an internal object, so the controller should go to the lexicon for its composer and then come back to handle whatever the composer replaced the element with.

However, there is always an intervening step to check for the possibility of pronominalizing. This check is made with the element still in its internal form. The record of the discourse is given directly in terms of the internal representation and test for prior occurence can be as simple as identity checks against a reference list, avoiding potentially intricate string matching operations with words.

In the dialog that this message came from, there is clear reference to <winston>, so it can be pronominalized and "he" is printed.

Any slot, or any node type may have procedures associated with it that are executed when the slot or node is reached during the second phase. These procedures will handle syntactic processes like agreement, rearangement of slots to realize features, add function words, watch scope relationships, and in particular, position the particle in verb-particle pairs.

Generally, particle position ("squeeze John in" vs. "squeeze in John") is not specified by the grammar - except when the object is a pronoun and the particle must be displaced. This, of course, will not be known untill after the verb group has been passed. To deal with this, a subroutine in the "when-entered" procedure of the verb group is activated by the "particle" procedure. First, it records the particle and removes it from the VG plan so it will not be generated automatically. A "hook" is available on any slot for a procedure which can be run after pronominalization is checked and before the composer is called (if it is to be called). The subroutine incorporates the particle into a standard procedure and places it on that hook for the object1 slot. The procedure will check if the object has been printed as a pronoun, and if so, prints out the particle (which is now in the proper displaced position). If the object wasn't pronominalized, then it does nothing, nothing has yet been printed beyond the verb group, and other heuristics will be free to apply to choose the proper position. Since <person being talked about> is here equal to the student, the person the program is talking with, it is realized as the pronoun "you" and the particle is displaced.

Going from <31-10-75,9am-12am> to "tomorrow morning" may be little more than table lookup by a "time" composer that has been designed to know the formats of the time expressions inside the scheduler.

This presentation has had to be unfortunately short for the amount of new material involved. A large number of interesting details and questions about the processing have had to be omitted. At the moment (September, 1975), the data and control structures mentioned have been fully implemented and tests are underway on gedanken data. Hopefully, by the end of 1975 the component will have a reasonable grammar and will be working with messages and lexicons form the two programs mentioned before. A MIT A.I. lab technical report describing this work in depth should be ready in the spring of next year.

David McDonald

David McDonald
Cambridge, Mass.

References cited in the text:

Genesereth, M. (1975) A MACSYMA Advisor. Project MAC, MIT, Cambridge, Mass.

Goldman, N. (1974) "Computer Generation of Natural Language from a Deep Conceptual Base". memo AIM-247, Stanford Artificial Intelligence Lab., Stanford, Calif.

Goldstein, I. (1975) "Barganing Between Goals". in the proceedings of IJCAI-4, available from the MIT AI lab.

McDonald, D. (1975) The Design of a Program for Generating Natural Language. unpublished Master's Thesis, MIT Dept. of Electical Engineering.

Simmons, R. (1973) "Semantic Networks: Their Computation and Use for Understanding English Sentences". in Schank and Colby eds. Computer Models of Thought and Language.

Winograd, T. (1972) Understanding Natural Language. Academic Press, New York, NY.