WORKING PAPER 76

conversations between programs

by

David D. McDonald

Massachusetts Institute of Technology

Artificial Intelligence Laboratory

September 1974

Abstract

This paper discusses the problem of getting a computer to speak, generating natural language that is appropriate to the situation and is what it wants to say. It describes, at a general level, a program which will embody a theory of how the various types of available information are used in the linguistic process as well as the possible packaging for some of that information and the experimental situation in which the program will be developed.

Work reported herein was conducted at the Artificial Intelligence Laboratory, a Massachusetts Institute of Technology research program supported in part by the Advanced Research Projects Agency of the Department of Defence and monitored by the Office of Naval Research under Contract number N00014-70-A-0362-0005.

This is a paper on speaking. More specifically, it is a paper on what is involved in arranging for a computer program to express itself in English. The ideas in this paper come from a line of research that I have been pursuing for some time and should find their way into a program to be built as part of my Master's thesis. This paper, however, will remain a long distance away from the implementation level and will let most of the potential detail remain fuzzy and unspecified.

1 A view from many miles up

I want to deal with what happens between the point when a program decides that it wants to say something, and when the English words have finished being printed out. Subsuming everything that goes on between these points under the label "linguistic process", we now have to determine: what things we could start with, what knowledge needs to be brought to bear, what could be the flow of control and interaction between the various sub-processes involved.

Postponing some preliminary answers to these questions until later, I want to initially discuss some of the higher level structure of the linguistic process.

This paper is a revision of my term paper for Professor Woods in 6.682, spring term, 1974.

It seems profitable to me to section the process into two realms which I will label composing and realizing. This formalizes my observation that while the linguistic processing is a unified procedure, there are two different sources of input and constraints, namely the speaker's own thought processes, and the state of ongoing discourse.

Composition is charged with deciphering the state of the speaker's head: coming up with words and syntax that could express what the speaker intends. Thus it combines a facility for interpreting the speaker, with generally applicable knowledge about words and syntax. The realizing realm worries about the nitty gritty of language production, things which are automatic to the native speaker: morphology, fast speech rules, trivial syntactic details and the like. Realizing is aware of the details and the import of the ongoing discourse situation: whether pronominalization is feasible at a given point, or how to flesh out the syntactic details that composition didn't care to worry about, such as perhaps focus, or default intonation contours.

What we have is a single process drawing on two sources of knowledge. Realizing and composing are two realms with *different points of view, maintaining environments for positioning rules and heuristics relevant to the decision making of the process as a whole.* Of course, to a certain extent their activities could be characterized as separate and connected by a pipeline operation; composition must do the first work toward an utterence while the final details will be filled in by realization. But this follows naturally from the linear, left to right organization of speech; the "division" between the two is purely conceptual. Control and motivation have the ability to move through the realms in whatever way is reasonable should the situation warrant it (unanticipated syntactic restrictions on some choosen word, or the insertion of a pronoun for example).

2) Some comparisons and a predjudice

To my knowledge, no one has ever written a system that included all the phases mentioned above: 1) primary program decides to speak 2a) The linguistic process uses the equivalent of composers to extract words and syntax options from the state of the program making the decisions required to specify the utterence, and 2b) the the equivalent of realizing processes take into account their specialized grammatical data to produce fluent English (or French, or Russian ....).

I know of only one system that addressed itself to composition (ERMA of Brown '73, and Clippinger '74). They did a fine investigation of the notions of examination, feedback, and and audience modeling, but they dealt with the meaning of their vocabulary and internal representations in an unanchored way. Their speaker, ERMA, is not a primary program. She has no existance in her own right apart from the linguisitic program and so there was no external source of justification for the motivation of her vocabulary. One has to accept the reasonableness of the way it is employed by the linguistic processor on faith.

Various Question Answering systems do, of course, "speak", but then so did ELIZA (Weizenbaum '66). ELIZA was never a real "speaker" simply because she did not understand what she said. If the user were to respond to everything that ERMA said with "T", she would never know the difference and would continue with her stock of pre-set replys (try it!).

Similarly, the Question Answering systems to date don't understand what they say either; they could never hold a true conversation. In all cases, their answering components are not much more than fancy collections of print statements that are not generalizable. Winograd's program ('72) does incorporate very reasonable heuristics for pronominalization which yield very natural sounding output. However, the system as a whole rests on a foundation of fill-in-the-blanks with some minimal, ad-hoc, object description which is totally fixed and cannot be expanded directly (see his thesis pp. 163-169 for further details).

Then there are a large number of systems that I would describe as only dealing with the realization process (Wilks '73, Goldman '74, Slocum '73, and possibly Heirdorn '73). They talk about generating language (surface structure) from (deep) semantic representations (some flavor of deep structure). As I see it, there is nothing terribly profound in their work, since realization is a very straight-forward process by comparison with composition and particularly since virtually the entire utterence is available at once in the semantic representation and all the complexity brought on by left-to-right generation is avoided.

The predjudice

I believe that there is a more serious problem with these last systems that will make it extremly difficult to eventually hook them up to intelligent primary programs (eg. a medical diagnostic program). It is, in essence, that I do not think the primary programs will find it convenient to package their output in the fashion that these programs can accept; assuming that they could do it at all without, in effect, becoming a linguistics program at the same time.

My reasoning is that the primary programs will employ many different kinds of internal representation, adapting appropriate representations to different internal structure (mostly for the convenience (and often the sanity) of their designers). I suspect that collecting these diverse forms into a commonly shaped package which is handed off to the linguistics, is more work than leaving them in their original form and giving the linguistics a model of how to find them, ie. supplying the linguistic process with some sort of brief guide to what you want done and why. This alternate technique is better because it explicitly removes a potential barrier in the carrying out of the linguistic process, namely that the feedback that must go on in deciding the structure of an utterence to be said, the processes involved must be able to have adequate information to smooth out the inevitable conflicts that will arise. Further, the conflicts may be so extreme that the primary program will have to change its mind about what it wanted to say and revise it to something more readily realizable.

Of course, this argument presumes a particular style of program design and may not hold for systems witten in different styles. In particular, a program based on a theorem prover would probably have no difficulties in keeping a single formalism linguistic component happy. However, it may be significant to note that one of the most sucessfull Question Answering systems to date, Winograd's SHRDLU, is a paradigm case of the sort of programming style I have in mind; time is seen by it as a list of past events, ongoing discourse is an A-list of noun groups with syntactic descriptions of where they came from, proper names are micro-planner theorems, and so on.

3) From whence cometh inspiration

One of my important guidelines while working on this theory is that ***the linguistic component should not formulate concepts or nuances that can not be found within the primary program itself.*** This is almost entirely a matter of keeping myself honest and goes along with my feelings about ELIZA, namely that a program that doesn't truely know what it is saying will never be able to carry on a conversation (I am reminded at this point of the conversations between the Doctor program and the Paranoid program, but I think they only make my position stronger). I also have a hunch that our use of language, the choices we make in specifying words and syntax, is deterministic; that we are not rolling dice in our heads (though there may be a lot of noise in our channels). Incorporating linguistic abilities into a program that didn't know what they meant does not go along with thinking that language is deterministic.

The most intimate (and most idiosyncraticly structured) interface between program and linguistics will be the vocabulary. In the initial stages of composition, each concept used by the primary program will point out a cluster of active routines and data that will, mediated by a model of the primary program, decide what lexical items are appropriate to express the concept. The clusters would include routines for choosing between alternatives, pointers to other parts of the model for filling in case frames, syntactic detail, and annotation to facilitate coordination with the rest of the linguistic apparatus.

Note that the linguistic process is never dealing with objects midway between those manipulated by the primary program and actual words (naturally when I use the term "word" I intend to include any syntactic or other data that is needed to deal with it as a linguistic object). In particular, this system will maintain no notion of primitive concepts apart from those that happen to be in the primary program as mediated by the model of it that the linguistic process uses. The linguistic component works directly at the level of words, except of course for the word choosing apparatus itself which I would describe as wired into the program and model itself and idiosyncratic to each primary program structure. My opinion on this matter of "primitive" concepts is about the same as Jerry Fodor's (personal communication '72): there are certainly as many "primitive" concepts in a person's head as there are words in their vocabulary and probably many many more, where by primitive, we mean dealt with directly rather than first being further broken down.

3a) A miniexample on the operation of the vocabulary

Consider a program with a model of chess. The model includes a concept that we can characterize loosely as "safety": the state of being protected from attack. As a linguistic description of this state, two phrases immediately suggest themselves: "(being) safe" and "(being/getting) out of danger". Notice that the "single" concept in the model maps into two elements in the vocabulary. Also notice that the elements are potentially multi-word - the "lexicon" as it were, incorporates idioms on an equal basis with single words, and it has its fingers on a lot of syntactic description: the internal structure of an idiom, restrictions on the permitted environment, variations for both idioms and single words, "case frames", etc..

The concept *safe* "points" to a single lexical entity "safe/out-of-danger" which is a very intricately structured beastie. *A lexical entity is partially an active program. In particular, it can thoroughly examine the state of the concept that nudged it in order to determine which lexical realization is appropriate to the situation.* It also includes a lot of information about how it interfaces with the more global linguistic situation, ie. "safe" nominalizes into "safety" and "out-of-danger" into "getting (or being) out of danger".

The criteria for choosing between "safe" and "out-of-danger" are very flimsy - the slightest contrary wind and they will collapse like card houses. On the one hand, the two are synonomous - if something is out of danger, it is also safe; why else would they be attached to the same concept in the model. But at the same time my native intuition feels very strongly that it would be wrong to describe a piece as "out of danger" if it had not actually been "in danger" before.

4) Some philosophy of Programming Style

Before turning to a discussion of composing and realizing, it would be better to describe my general biases about how things go on. This phenomenon of syntactic forms that are synonomous but not entirely, or of words that are often used but hard to precisely define ("a brilliant move") is rampant in language as we use it and I see no way to deal with it except via very loose structures and light, tentatively acting formalisms (formalisms that readily permit a decision to be reconsidered after it has been made).

I want to suggest that human linguistic decision making and the linguistic process as a whole are characterized by fuzzy motives and inconsistency of behavior over time. In light of this, it seems extremely doubtful that fluency can be reasonably described by monolithic, straight-line programs (such as SHRDLU's grammar) - certainly not by readable ones in any case, because they cannot maintain the required flexibility and the required perspicuity at the same time. More reasonable to me are programs that factor their knowledge into small bodies; small active specialists with a great deal of structure - programs that know a great deal about each other and about the structure and flow of the process that they are imbedded in. Further, I do not think that "fuzzy" linguistic decisions should be handled by fuzzy program interaction such as using blind pattern-directed invocation any time that you need something done, letting whoever you come up with use their technique of the day, or using simulated dice to settle syntactic choices. Instead, I see the proper model for fuzzy thinking to lie in increasing the level of interaction between specialists so that responsibility for a decision is widely spread and so that decisions can be countermanded when more compelling motivations arise (though not without the "approval" of the program that originally made the decision). Also, there will have to be copious annotation of the authors and motivations connected with each decision so that the programs will have something to talk about when joint decisions are required.

5) An example of a specialist

To expand on this notion of "interaction between specialists", let me go into some detail on one particular linguistic fact and how it could be incorporated. Actually it's a very flimsy "fact" since it lies in an area of high variation within syntax. A better characterization would be to say that this piece of knowledge is a "heuristic for choosing style"; using it might make what you say "sound better" or might stave off some potential ambiguity. The fact has to do with quantifier scope and with the explicit repetition of structure across a conjunction. Consider this example.

" ...showing that certain mechanisms that neurophysiologists propose are not well defined or inadequate to carry out the behavior they are supposed to account for."

J. McCarthy
('74 paper, not yet published)

The sentence as written makes instant sense only if the words "not well defined" are intoned as one word with falling intonation and "inadequate is given sufficient prominence. Otherwise (the normal situation), the "not" is interpreted as belonging with "are" and carries across the conjunction yielding "or (are not) inadequate". Of course humans throw this possibility out because they know what McCarthy intends to say - but none the less, the sentence "sounds bad" to many readers. The stylistic hack for preventing this is to repeat the verb (actually this applies more generally) on the other side of the conjunction, signaling that the scope of the "not" is restricted to the NP that it was with.

How do we encode this heuristic? Presumably it will exist as a specialist, but what does that mean? First we should consider when the need for it will come up. That will be just when we know that we will be expressing something via a conjoined structure, which immediatly follows a verb that can assimilate a "not". I see this as a matter for realization rather than composition, based on very subjective judgements about how much of a syntactic flavor the heuristic has and its "distance" from meaning.

Because we are realizing from left to right as material becomes available, it is reasonable to assume that at the point of realizing the verb, we could have been warned to expect a conjoined structure without knowing anything about the shape of its components. The warning calls in our heuristic which checks the verb and if it fits, proceeds to perch at the level of the verb and the conjoined structure marker, waiting to see the shape of what will lie in the first phrase of the conjunction. If it begins with a "not", then the heuristic becomes nervous and waits expectantly for the next phrase to come in. Should the second clause either not have a "neg" on it, or have one in a form that is unsuitable for assimilation by the verb (as in the example above), then the heuristic takes corrective action to insure clarity, and arranges to repeat the verb just after the conjunction has been realized.

6) A lengthy example of the complete process

Let me try to explain the action of composition and realization via a rather extended example drawn from the scenario of my thesis research. The discourse situation I will be dealing with is the conversation of a collection of expert routines that contemplate chess board situations; their problem is much the same as the one in the protocols taken by Newell and Simon ('72) They perform their analysis by each doing what they are best at, and communicate between themselves via messages. The flow of messages constitutes a sort of conversation and the action of the linguistic machinery is to "translate" those messages, as they occur, into fluent English. The translation is done purely as an excuse for language generation since the programs themselves send and receive only messages in an internal format.

The question of deep structure/semantic representation is important to consider here. It is tempting to think of the "messages" being passed around as the underlying representation of the English utterances they are translated into. Such a thought is dangerously inaccurate and could lead to a considerable misunderstanding of what is actually going on and what constitutes the meaning of an utterence in this kind of situation.

Bear in mind that what we have are programs who know the relevant parts of each other's structure very well, and who can therefore talk via very idiosyncratic messages. Certainly the probable structure of the messages will bear no resemblance to any known family of linguistic descriptions. A program will be free to send any structure that the receiver can decode. Typically a message could consist of an atom created for the occassion, with information on its property list in a manner peculiar to the situation involved, possibly including: symbols, code, code macros, lists of symbols, board states, etc..

In some sense, the meaning of the final utterances is the physical message passed. If the receiver can decode it, a linguistic routine (a composer usually) can do the same, using the technique of the receiver as a guide, however, ***the information that can be gleaned from the message*** *is inadequate to specify the English utterance.*

There are natural, fundamental differences between the organization of a program and of a language, which will dictate that the program's linguistic elements must search over area of the program/model beyond the message, in order to get enough information to realize a given situation. In particular, the elements of a linguistic description are distributed through time; something is said first, something second, etc., whereas all aspects of a description in the program/model exist simultaneously. The program/model thinks in simultaneous terms and there is nothing sacred about order because the "arguments" to a predicate are randomly accessed by name.

Ordering information must be found elsewhere than in the basic predicate instantiations contained in the messages. Some predicates will map into verbs with a partially fixed order because of the contrasting semantic natures of their arguments (a rank is seldom an agent), but something like "attack" or "defend" clearly requires additional information on which to base an ordering decision which will subsequently be realized (usually) as passive vs. active (the knight attacks the pawn - the pawn is attacked by the knight).

This kind of information - usually called topic/comment, stress, focus, information flow (all very ephemeral notions at the moment) must be found elsewhere in the state of the program/model as it exists at the time of the realization. The previous discourse has set up a pattern that the new utterance must be consistent with and the composers will have to examine speaker and hearer for their motives, biases, preferences, as necessary.

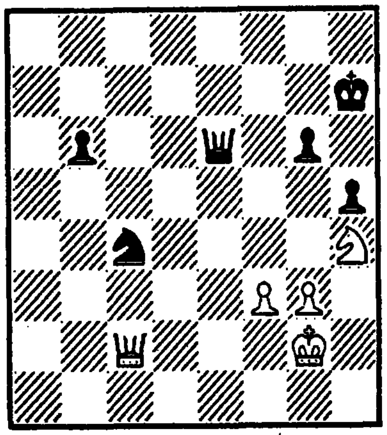
--- A significant feature of this microworld

Unlike discourse situations that have been used by researchers in the past where the interaction has always been between program and human, this scenario involves two or more programs. This is tremendously beneficial for studying the plausible heuristics which might govern these little understood linguistic processes. The entire state of the participants is available if needed, and most importantly everyone is working at the same level of complexity. In the usual case of programs talking to humans, the program must consider the human to be essentially a black box because the generalization that would make human behavior consistent are virtually beyond the realm of feasibility for the program given the present state of the art. By dealing with program/program discourse, I hope to have narrowed the problem sufficiently to allow significant progress in reasonable time.

But enough of the preliminaries, let me give you an example, perhaps too long, of some hypothetical composers and realizers at work on a specific message.

--- The example

The board position from which the scenerio is drawn.



position 100, from Horowitz (1950)

White to move.

The scenario is as follows. Given a board situation, one expert, very much aware that the side it roots for is down a pawn, has noticed that an enemy pawn can be taken on the next move and is lobbying very strongly for NxP as the next move. A special subcommittee is checking this out to see if it puts them in a worse way several moves later on. Someone notes that a possible reply to NxP is QxN and wants to check if the opposition is prepared to make this move and if so, where things would stand afterward. A program is delegated to pretend that it were on the other side and ask if anyone would mind if the Queen were moved from where it is. It happens that the Queen is involved in guarding a piece under attack on another part of the board. This situation is embodied in an active program that responds when the delegated program asks "would anyone mind if...", and it will·then send a message replying to the question. Let us examine what that response looks like and how it might be realized in English.

We can say for our purposes here that the structure of the message looks like this (Capital letters are LISP atoms, and *italics* are properties on atoms. When atoms are mentioned, they should be viewed as a collective name for more complex pieces of information that will be found on the atom's property list along with instructions on how it is to be interpreted.).

```
MESSAGE-22 type - comment
           value - nix
           on - OPENMOVE-2
                              value -a complex data structure
                                     to the effect that the
                                     Queen moved off DIAGONAL-2"
           reason - REASON-12
                              type - consequence
                              following-from - MOVE-43
                              value - EVENT-120
```

(I hope that it is clear that something like this does not fit into the format of your garden variety underlying representation.)

The message is intercepted by the linguistics at the receiver's end, where the annotation within the receiver can be used to aid in its interpretation. The first composer to get a crack at the message is a general-purpose dispatcher. It looks at the properties on the message, using the receiver's annotation if necessary, and figures out what kind of message we have and what composer(s) it should be given to.

The message is of *type* comment which is an indicaton of its relationship to the on-going discourse (it is commenting on something done/said earlier). This fact will be of interest to the realizing routines who maintain the discourse context and so it is shipped out to them along with the information that we are starting a new utterance. The dispatcher fingers the *value* property of the message "to see what it says". The *value* is a concept or knowledge bundle of some sort and as such it includes a composing program that the dispatcher activates.

This composer, we can call it NIX, uses a general format (for its compositions) of "(imperative) <neg - action determined by the message>, <additional material as the message requires>". This determines the high level ordering and specifies (subject to revision) the syntactic shape of the first clause. The format is passed to realizing to let it know what to expect later in the way of structure and what constraints to be on the lookout for.

Composition proceeds from left to right after an ordering has been imposed on the message. But before actually composing anything, a quick check is made to determine the nature of that "additional material" if any. There is potentially reason to modify the normal ordering depending on what the composer finds here, but in this case, NIX observes that the only relevant thing on the message is the property *reason* which it usually wants to realize as a bound clause. Naturally it tells this to realizing so that it knows how the utterance is expected to finish up. Preliminaries over with, NIX gets on with the business of composing. The action (found under *on*) is an instance of a particular abstract action known to the program/model, namely "moving off this particular diagonal". The action is of a type that employs verbs with the auxiliary "do", neg is part of the auxiliary syntactically, and so NIX composes the structure "(AUX (NEG do))" and sends it off to be realized. The "do" in the structure is a lexical item, and not something more abstract.

Realizing hands off the structure to its "aux" expert and attaches it leftmost under the S-node previously set up. The aux expert does its ordering, and gets morphological information to indicate that it should employ a contraction, yielding "don't". Now realizing is aware that what follows is of type action and that the utterance is type comment. One usually comments only on things mentioned before, therefore there will probably be an antecedent somewhere in the preceding discourse ***and we can pronominalize the action.*** So it asks NIX for the internal pointer to the action (OPENMOVE-2) and goes searching among candidates for antecedents looking for sufficient similarity to justify getting away with a pronoun here. Let us assume that it finds one, and so, depending on its location and nature, realizing inserts the appropriate pronoun, employing a pronoun specialist to manage the details. In this case the proper choice is "do that".

All that accomplished, it proceeds to tell NIX to forget about coming up with a composition for OPENMOVE-2 and to move on to the reason instead (realizing also puts out a comma at this point since it knows that the bound clause is coming).

To compose the reason, NIX first inquires of its value what kind of syntactic form it is likely to have (this kind of information is known by the consequent composer). We don't want to have to wander into the composition of the specific situation just yet, because in its present state ("<imperative main clause> , with a reason left to compose"), NIX would like to use the binder "because" which may not fit with what follows, hence the check against potential syntactic forms.

--- **to interupt for a moment**

It is most likely that we will eventually be looking for far more subtle things than "potential syntactic forms"; both for this problem of whether or not to explicitly realize the binder "because", and for deciding many more fuzzy linguistic choices. ***There is not enough known at this time about the structure or workings of English to motivate the choice, by a speaker, of even the simplest of linguistic forms.*** We can formally spell out what choices may exist, but we have only the vaguest notion of what criteria decide between them in a given situation.

This puts the researcher who wants to write a speaking program today in a bind. How can she be anything but arbitrary about the decision criteria she writes into the program's grammar? For example, the problem here of including or not including "because" is essentially the same problem as deciding whether to include the relative word with a relative clause (nb. WHIS deletion), which I suspect we have even fewer guesses about. The phenomena pervades linguistics and is compounded because the human data is complex and inconsistent.

Since I am presuming that the problem will not be miraculously solved before I finish this work, I have decided on two courses of action for dealing with it as it arises in this context. The first is avoidance; the chess world and the discourse of its programs will not call for many of the more problematical decisions (ie. subordinate clauses) because of the vocabulary it involves. The second is simply that the entire system ought to be seen as an experimental scenario. If the given heuristics have the programs producing natural sounding language, then they may be correct. If they do not, then the mechanics of changing them should be simple so that we can experiment, developing a laboratory for determining what the proper ones might be; refining our initial crude guesses into something nearer to the truth.

--- Back to the example

The insertion of "because" at the start of the bound clause could, it seems to me, serve two discernable functions. First it says that what follows is the reason for what went before; however, this may be perfectly obvious from the words in the clause itself (explicit vs. implicit presentation). Second, including the word smooths out the flow across the clause boundary. There are no pauses. Assuming that this is a good thing, we could assert (not having any better ideas) that including "because" should be the default case. With that as a flimsy guideline, we can now let NIX ship out "because" and consider at some other time what the total effect sounds like.

At this point, NIX has given control to the concept it found on the *type* property of its *reason*. As we saw on the message, consequents follow from something and have a value, in this case an EVENT-.

EVENT-122 *value* - (allowed MOVE-50)
BY OPENMOVE-2

A composer of a consequence wants to compose its value and to make sure that it is clear what it is a consequence of. What toplevel organizations, akin to the one used by NIX, could accomodate such a description, and how can the composer decide between them? Barring compelling reasons from discourse (which I don't believe would show up in this micro-world), the basic order should be "<reference to cause>,<description of consequence>". The reference-to-cause might be more or less pronominalized depending on how well established it already is in the discourse. While the number of potential variations on that theme is unfortunately large when viewed at a sufficiently detailed level, I suspect they will all fall into two variations: "cause-reference + full clause", or "nominalized reference + verb phrase". Typical examples would be ",because then ..." vs. ",because it <VP>".

Of course there are numerous other choices (",because if you do that ..."), but the nuances that distinguish them may not make sense to the program/model and at the present, researchers have no idea of whether humans consistently notice them, or what they are doing if they're not using them. It is important to anticipate dealing with the nuances of constructions, but foolish to insist on perfect lucidity and fidelity in programs under development.

The choice between the two patterns given above as available to the composer is based on syntactic considerations. Choice one, "because then, ...", describes the consequence via a full clause while choice two, "because it ...", subsumes the cause into the clause as its subject and describes the consequence as a verb phrase. ***This will only be possible for some consequents.*** In the present case, the consequence predicate is "allowed" and its "subject" is precisely the cause of the consequence, indicating that the second form is the one to use. Naturally, the pronominalization had to be explicitly checked out and the pronoun chosen and so on, but we can assume this sort of detail for the rest of the example.

At this point, the composer intends to use the predicate "allowed" as the main verb. Predicative concepts are not automatically verbs; they keep around a description of their possible and prefered syntactic/lexical forms which composers have to check against, as this one did in deciding to use the second form. Realizing has established an environment for this clause and so it specifies that the precise form of the verb that gets realized will be the present singular, "allows". Next comes the composition of the argument to the predicate, MOVE-50, which is arranged for under the direction of the main verb (syntactic interfacing decisions within the clause are controlled by the verb and largely independent of the nature of its argument though the argument may rule out certain possible compositions).

At the time of the decision to use this pattern, realizing had the structure "(s *)" pending below "because" (the asterisk is the point at which the next structure coming down the line will be added unless instructions are received to the contrary). After making the pronoun choice the structure is "(s (np it)(vp *))" - or some such, the details are not yet clear. This syntactic tree is not maintained merely for aesthetics; it is a structuring of the realization environment that provides hooks for the watchdog routines that are potentially brought in with any of the nodes. The dictates of the MVB on its arguments are actualized via this structure.

The essence of MOVE-50 is that the formerly protected knight is exposed to immediate attack from the opposing Queen. Knowing that MOVE-50 is a chess move, the composer can call in the expertise of a general move-composer. Let us say two verbs are available to the move-composer to express the situation: "attack" and "take". There is also the choice of passive vs. active to decide. Of course, this is chess. It might be most appropriate to use chess notation, QxP, and ignore the subtler questions involved with the possible English questions. On the other hand, using chess notation would be an arbitrary decision, apart from anything in the program/model - essentially turning a switch in the linguistics component - it is better that we consider the full process here.

"Attack" and "take" must maintain, in their lexical clusters, material for choosing between them. Perhaps we could say that "attack" allows for counterattack, regaining the losses, while "take" has a greater sense of finality about the loss. The program that is the speaker is sufficiently sophisticated to make such a distinction, and for this situation, the proper sense is "take" (I summarily wave my hands over the details of the decision). The active voice is preferred in English, but the item of importance to the speaker, the Knight, is also usually fronted, indicating the passive voice. Fortunately, in this case at least, the situation is more complicated and there is a way out of the dilemma. MOVE-50 is being realized as the complement to "allow". This is bad for the passive realization since the fronting achieved is not actually to the front of the utterence or even the front of a toplevel clause. Further, the fact that the verb is "allow"and that MOVE-50 is conceptually timeless dictates that the composed clause should be infinitival. Passivized infinitives are awkward sounding at best. With our knowledge of language what it is, we can not afford to look for any more principled criteria so this should certainly be enough evidence to indicate using the active ordering over the passive.

Describing chess pieces is a simple straight-forward process; they are named, and the dissambiguation process follows a well defined and easily understood (to the program) pattern. The details of the opperation are irrelevant to the current example. From left to right, the general move composer puts together the combinations of words and ships them off to be fitted in and realized. The verb can be shipped out as a simple lexical root form since its syntactic form was entirely dictated by the higher verb and the detail will be provided by realizing.

At this point, we have exausted the information in the message and all the active composers have finished what they wanted to do. In short, we are finished and the linguistic component has come up this the sentence: "Don't do that, because it allows the Queen to take the

Knight".

8) Theoretical considerations

A theory of language embodies our answers to what we think the important questions are. For me, these questions include: why do we speak the way we do; what is the relationship between language (particularly words) and meaning; what are the constraints and generalizations that govern our speach; what is language such that we can learn it, be aware of it, have intuitions about it, and so on.

Given what you want to ask, there is the problem of how to approach the data. What do you look at and what questions do you ask of it? In any science that deals with observations in the real world, good practice requires you to separate signal from noise when looking at data. In language, coughs, stuttering, backing up, spoonerisms, and forgetting where you were are all noise. Linguists call them all aspects of "performance". This is as opposed to "competence", a model of what a speaker is competent to produce, all other things being equal (Chomsky's notion of "competence" is only one possible formalism among many when we emphasise the original motivations for the distinction).

With the criteria stated as loosely as they are, most any sort of schema could be used in a competence model. After surgically removing the micro-world from my own system, you would be left with a "competence" model which would, in effect, map situations in a hypothetical program/model into finished English utterances. Furthermore, the model would be novel in that it explicitly includes the points of linkage between linguistic competence and knowledge of the real world and of the situation at hand. Given those links, the theory can address itself to the situational aspects of language, why a particular utterance was used in some particular circumstances.

I think this is the most important aspect of the system that I am developing, that it includes connections between the linguistic process and the situation at large. The lexical clusters, the discourse realm in realizing, and the left to right production are all points of connection. With the additional, real world constraints that can be drawn from these phenomena and others, I think it may be possible to approach the questions we have about language with some hope of finding satisfactory answers.

BIBLIOGRAPHY

Brown, Richard (1973) *Use of Multibodied Interupts in Discourse Generation*, unpublished Bachelor's Thesis, Dept. of Electrical Engineering, MIT.

Clippinger, John (1974) A Discourse Speaking Program as a Preliminary Theory of Discourse Behavior and a Limited Theory of Psychoanalytic Discourse unpublished Ph.D Thesis, Dept. of Linguistics, University of Pennsylvania.

Goldman, Neil (1974) *Computer Generation of Natural Language from a Deep Conceptual Base*, Instituto per gli Semantici e Cognitivi, Castagnola, Switzerland.

Heidorn, G. E. (1973) *Simulation Programming Through Natural Language Dialogue*, IBM recearch report RC 4535, Yorktown Heights, New York.

Horowitz, I. A. (1950) *Chess for beginners*, Simon and Schuster, New York, N.Y.

Newell, Allen and Simon Herbert A. (1972) *Human Problem Solving*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Slocum, Jonathan (1973) Question Answering via Cannonical Verbs and Semantic Models: Generating English from the Model, Technical Report NL 13, Dept. of Computer Science, Universtiy of Texas.

Weizenbaum, J., (1966) ELIZA. Communications of the Association for Computing Machinery, 9, 36-45

Wilks, Yorick (1973) An Artificial Intelligence Approach to Machine Translation, in Colby and Shank eds., *Computer Models of Thought and Language*, W.H. Freeman and Co., San Francisco.

Winograd, Terry, (1972) Understanding Natural Language, in *Cognitive Psychology*, 3,1 1-191.