Subsequent Reference: Syntactic and Rhetorical Constraints

David D. McDonald
MIT Artificial Intelligence Laboratory
Cambridge, Massachusetts 02139

Abstract

Once an object is introduced into a discourse, the form of subsequent references to it are strongly governed by convention. This paper discusses how those conventions can be represented for use by a generation facility. A multistage representation is used, allowing decisions to be made when and where the information is available. It is suggested that a specification of rhetorical structure of the intended message should be included with the present syntactic one, and the conventions eventually reformulated in terms of it.

Introduction

Whenever a speaker wants to refer in text or speech to some object, action, state, etc., she must find phrase which will both provide an adequate description and fit the context. What governs her choice? One way to find out might be to look at the selected phrase after the fact and try to develop a static characterization of the relation between it and its context. This is what most non-computational linguists do. However, relations derived from finished texts are at best incomplete. They will not tell us how the choice was made or even guarentee that the relation(s) was apparent when the choice had to be made.

To get a clear picture of what people know about making references, we have to focus our attention of the process that they go through. It must involve making decisions on the basis of some contextual evidence. What is the evidence? How and when is it computed? How is it described? Is the decision of what phrase to use made all at once or as a gradual refinement? How is this process interleaved with the larger process of constructing the rest of the utterance?

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defence under Office of Naval Research contract N00014-75-C-0643.

We can narrow the research problem by distinguishing two kinds of references: initial and subsequent. This classification divides instances of reference by their position in a discourse. "Initial" references introduce new entities into the discourse, while "subsequent" references are another mention of one already introduced.

An initial reference must be an encompassing enough description of the new entity that the audence will be able to recognize it. This requires matching goals with evidence from a model of what the audience is likely to already know and how likely they are to understand various choices of wording (e.g. which of its properties should be emphasized? - why is it being introduced?). This is not easy. People talking or writing about unfamiliar things or to unfamiliar audiences are not particularly good at it.

Subsequent references are another matter. They are very highly grammaticized. While an initial reference may take almost any form: noun phrases with unrestricted numbers of adjectives and qualifying phrases, nominalized clauses, verb phrases (for actions), etc., subsequent references must use very specialized forms: personal, reflexive, and personal pronouns; special determiners like "*this*" or "*my*"; class nouns like "*thing*" or "*one*"; and so on. Here, grammatical convention dictates most decisions and leaves only some details to free choice.

* * *

My observations in this paper are based on experiences with a program for generating English texts from the goal-oriented, internally represented messages of other programs. My program, and the state of the art in general, can deal much better with the representation of a grammar than with then representation of an audience model. Hence the focus here on subsequent references.

The next section looks at the course of the whole generation process as my program models it, and fits the sub-process of finding phrases for references within it. Then the process of deciding whether or not to use a pronoun will be examined in some detail. This will lead to the problem of accessing audience models and, the idea that the relevant information should be computed outside the linguistic construction process per se. That idea is expanded to include "rhetorical structures" like the relation "all of a set" that leads to a phrases like "*...a square, ...the other square*". Finally, a design for this rhetorical structure is sketched.

Internal representation

Suppose we had a logically minded program that wanted to make the statement:

$$\forall x\ man(x) \rightarrow mortal(x)$$

People who have worked on language generation have almost universally factored out all of the program's knowledge of language into a temporally and computationally distinct component. Once the rest of the program has compiled a description of what it wants to say like the formula above - it passes it off to its "linguistic generation component" and lets it come up with the actual text.

But before moving on to that component, let us look closer at this formula. I am presuming that the speaker's primary (non-linguistic) representation, be it predicate logic, semantic nets, or whatever, uses a totally unambiguous style of representation - something equivalent to always refering to an object, etc by its unique name For example, the three "x"'s in the formula all denote the same object (albeit local). The two predicates, the quantifier and the implication sign all denote different objects.

We usually think of objects - noun phrases - as being the only things that might be refered to more than once, but that is not the case. Consider the formula **mortal(Romeo) ∧ mortal(Juliet)**. That could be rendered in any of several ways including: "*Romeo is mortal and so is Juliet*". Here the second instance of **mortal()** was realized by a special, highly restricted grammatic device - exactly the characteristics of a "subsequent reference". From the point of view of the language generation component, the important thing will be the repetition of some name from the input formula not, at first glance at least, the kind of object that name denotes. (The set of descriptive formulas supplied to the linguistics component is called the program's "message" Subformulas or terms within a message are called "elements" or "msg-elmts".)

The internal objects that appear in a speaker's descriptions will have defining and incidental properties associated with them which are accessible through their names. This will include a property (actually a packet of properties and procedures) which records what the program knows about realizing the object as an English phrase. I refer to this property as the object's "entry" - as in an entry in a translating dictionary. An entry specifies what are the set of possible English phrases that could be used for the object, and includes a set of context sensitive tests that will indicate which phrase to choose. Breaking down the speaker's "how to say it" knowledge into such small chunks facilitates the use of a general recursive process for turning messages into texts by following the compositional structure of the formula(s) from top to bottom.

Besides pointing to permanent properties, a object's name will also be the repository of more or less temporary annotations. In particular, when the generation component realizes an instance of an object as phrase, it can add an annotation to it marking what kind of phrase was selected, where in the text this occured, what the immediately dominating clause was at the time, and so on. The next time there is an instance of that same object the annotation can be found and used to help decide what kind of subsequent reference should be made.

Before the linguistic processing is begun, is it possible to examine the input formula and determine what subsequent references it will educe? The bound variable x appears three times, once with the quantifier and once with each predicate. It would be a candidate for some subsequent references if, in fact, the formula was rendered into English literally.

"*For any thing, if that thing is a man, then it is mortal.*"

But other, more fluent, renderings of that formula will not give the x's a separate status:

"*Being a man implies being mortal*"

"*All men are mortal*"

In short, it is not possible to predict which objects will be explicitly refered to and which not just on the basis of a formula in the internal representation language. You would have to know (1) how the terms that dominate the object in the formula are going to be rendered; and (2) whether the object was mentioned earlier in the discourse and how it was described there. Then you would still have to, in effect, duplicate the reasoning process that the generation component would go through itself.

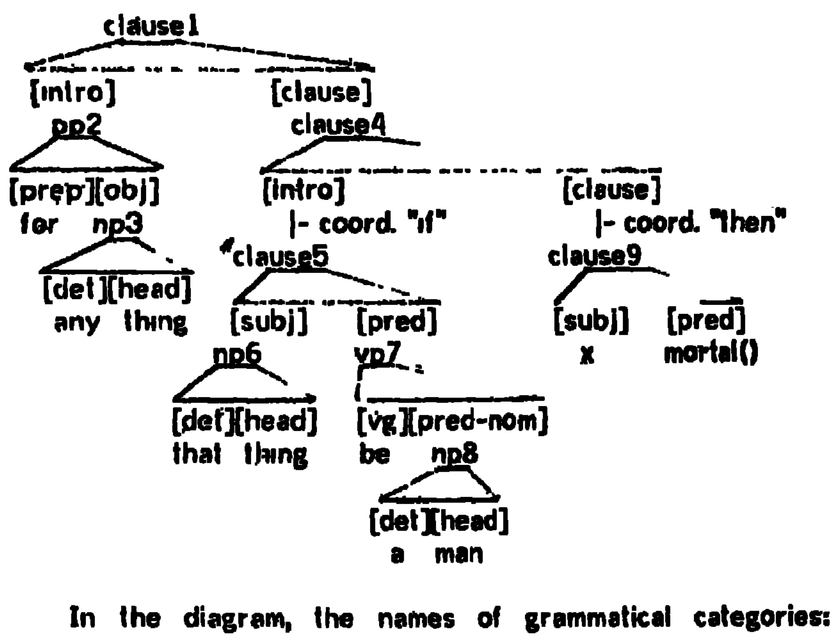
A we will see later, the generation component will often need "advice" as to whether or not the audience would understand certain phrasings. The audience model which makes these decisions will presumably prefer to work from pre calculated observations so as to avoid delay. The implication of the fact that you cannot whether that there will be a subsequent reference to a particular object until it actually happens is that you cannot make special preparations for it The audience model, or any other effected part of the program, will have to be generally prepared for whatever objects might be asked about.

The possibility of three different renderings for the same formula implies that the formula *per se* does not contain enough specification to pick out just one of them. If you consider the three sentences for a moment, you will appreciate that what distinguishes them are differences in rhetorical emphasis and in how to interpret ∀x. These are things that Frege deliberately omitted from the predicate calculus. To direct the generation component so as to arrive at a particular one of those sentences, more formulas would have to be added to the message or else found in the larger context (e.g. the formula might be part of a proof), and the entries for quantifiers, implication, etc. would have to be augmented to notice them.

Upgrading the predicate calculus enough to motivate the use of fluent English is a facinating problem, but one which I will gloss over in this paper. See McDonald [1978a] for more details. For now, I will assume that the decisions made by the various entries come out so as to give the literal version of the formula with the explicit references just so that we can use it for an example.

Syntactic Context

Below is my program's representation of the situation just as it is about to choose a phrase for the third instance of x in the formula. The point of showing this constituent structure is to demonstrate that while the program has a great deal of data to bring to bear on the choice, it also has a great deal of data which is utterly irrelevant to it. The packaging of the data - the size of the search space - is at least as important as having the data available in the first place.



In the diagram, the names of grammatical categories: clause1, pp, etc, denote the syntactic nodes of an annotated surface structure. Each node has a set of immediate constituents, organized by a list of named constituent slots. A slot can be empty, hold another node, hold a word or idiom, or hold an element of the input formula which has yet to be processed, e.g. x, or mortal(). The words at the leaves of the tree are given in their root form. A morphology subroutine specializes them for number, tense, etc. when they are spoken (printed on the console).

The choice of what syntactic categories, descriptive features and constituent slots to maintain is tied up with the choice of actions associated with them by the linguistics component. The [intro] constituent, for example, will act to insure that any introductory clause is realized as a participle. There are many trade-offs involved in the design of this grammar, and I will again gloss over them for this paper.

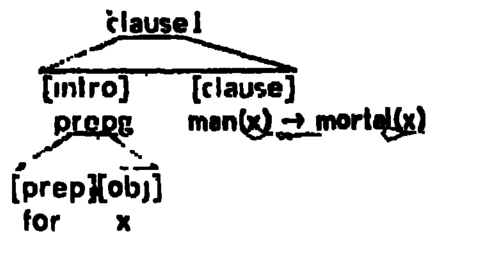
The choice of refering phrase for a subsequent reference is determined largely by the syntactic relationship between the current instance and the previous instance to the same object. In a static, after the fact analysis, we would determine this relationship by examining their positions in a tree like the one above. This is a simple enough operation for a person using her eyes, but it is an awkward mark and sweep style search for a computer program.

My program uses a much more efficient, and I would say more perspicuous approach based on recording potentially relevant facts at the time they are first noticed by the linguistics component. The wording of the heuristics that are used for the decisions are similar to the wordings used in static analysis. (They almost have to be, given that that is how the bulk of linguistic research has been done to date.) But the data for the heuristics is acquired in a more natural manner.

Before discussing the program actual pronominalization heuristics, I will first digress to describe the workings of the generation process which collects (and creates) the data.

* * *

The tree in the previous column was developed incrementally. Clause1 is the result of realizing the conceptually topmost part of the input formula - the quantification. Its argument - the implication - was then positioned in the new syntactic structure but not yet realized itself. This is what the constituent tree looked like at that point.



All of the generation component's actual knowledge is spread about many small, local routines: dictionary entries for the object that will appear in input formulas; "realization strategies" - the construction routines that those entries execute to implement their decisions; or "grammar routines" -

associated with the names of categories or constituents and in charge of effecting conventional details not involved in conveying meaning These routines are all activated and organized by a simple controller.

The controller works by walking the constituent tree, top down through the syntactic nodes and from left to right at each level of constituents, The process begins with the top node of the tree just after it is built by the entry for the the topmost element of the input formula.

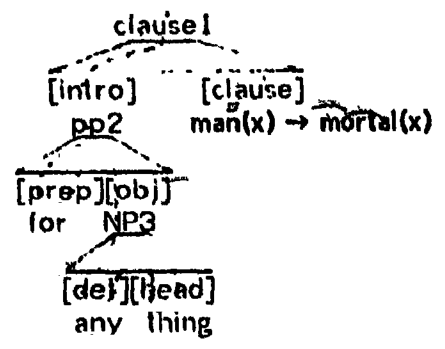
Outline of the Controller

```
Examine-node
    (1) call the grammar routine for this category node
    (2) rebind the node recursive state variables
    (3) call Examine-constituents

Examine-constituents
  - For each constituent slots of the current node in order do:
      (1) call the grammar routine for that slot name
      (2) call Examine-slot-contents

Examine-slot-contents
  - Cases:
    contents = nil        do nothing
    contents = <word>
       call the morphology subroutine with the word
       print the result
    contents = <node>
       call Examine-node
    contents = <msg-elmt>
       use the dictionary entry for the element to find
       a phrase for the element; replace the element with
       that phrase as the contents of the slot;
       loop through the cases again.
```

So, having generated clause2, in effect by starting the controller on the last case of Examine-slot-contents, the controller will loop around. The contents will now be clause2; the third case will be taken and the clause "entered". Its first constituent contains another node; the controller recursively re-enters Examine-node and enters the prepositional phrase. Its first constituent contains the word "*for*", which is immediatedly printed out with no changes from the morphology subroutine; the second contains the first instance of x which is processed with the dictionary entry common to "issolated variables". The noun phrase it constructs replaces the x in the constituent tree; the controller then loops through the cases once more, recursively calling Examine-node on NP3. It is now three invocations deep. The dotted line shows its path.



spoken: *"For any thing, "*

After processing np3, the controller will leave the np and thepp, go to the next constituent of clause1, use the dictionary entry for implications, and so on, et cettera.

The design of this generation component is oriented around the decision making process of the dictionary entries (see [McDonald 1978b] for more discussion). The principle reason that the process is deterministic and indelible, for example, is to simplify the conditions that the entries will have to test for. A more relevant example here is the use the controller to "pre-calculate" certain relations about the context and make them available through the values of recursive state variables maintained by Examine-node. For example, the controller keeps pointers to the "current-main-clause", "current-verb-phrase", etc.. It keeps track of whether it is in a subordinate context, of what the last constituent was, last sentence, and so on.

Any of these relations could be calculated independantly by directly examining the form of the constituent tree and the annotations on its nodes and embedded message elements. But the point is more than just efficiency. By making certain relations readily available and not others, one says that just those relations are the important ones for making linguistic decisions. A one of a kind operation like subject-verb agreement will have a special predicate written for it that "knows" where to find the relevant subject constituent in the constituent tree. But relations that are often used, particularly those needed for evaluating pronominalization, are maintained by the controller, and, as a corollary, are only available in their pre-computed form when the controller is present at that point in the tree.

The design of the controller guarentees that the generation process will have these properties: (1) It is done in one pass - the controller never backs up. (2) Therefore decisions, choices of phrasing, must be made correctly the first time. (3) It is incremental. When the first part of the text is being printed out, later parts will be in their internal form. (4) Therefore very specific facts about the linguistic characteristics of earlier parts of the text are available to influence the decisions made about the later parts. (5) In particular, when the time comes to render any particular associated with the names of categories or constituents and in charge of effecting conventional details not involved in conveying meaning. These routines are all activated and organized by a simple controller.

The controller works by walking the constituent tree, top down through the syntactic nodes and from left to right at each level of constituents, The process begins with the top node of the tree just after it is built by the entry for the-the topmost element of the input formula.

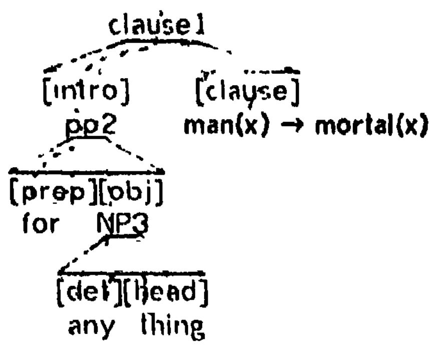
<u>Outline of the Controller</u>

```
Examine-node
    (1) call the grammar routine for this category node
    (2) rebind the node recursive state variables
    (3) call Examine-constituents

Examine-constituents
  - For each constituent slots of the current node in order do:
      (1) call the grammar routine for that slot name
      (2) call Examine-slot-contents

Examine-slot-contents
  - Cases:
      contents = nil        do nothing
      contents = <word>
         call the morphology subroutine with the word
         print the result
      contents = <node>
         call Examine-node
      contents = <msg-elmt>
         use the dictionary entry for the element to find
         a phrase for the element; replace the element with
         that phrase as the contents of the slot;
         loop through the cases again.
```

So, having generated **clause2**, in effect by starting the controller on the last case of Examine-slot-contents, the controller will loop around. The contents will now be **clause2**; the third case will be taken and the clause "entered". Its first constituent contains another node; the controller recursively re-enters Examine-node and enters the prepositional phrase. Its first constituent contains the word *"for"*, which is immediately printed out with no changes from the morphology subroutine, the second contains the first instance of x which is processed with the dictionary entry common to "issolated variables". The noun phrase it constructs replaces the x in the constituent tree; the controller then loops through the cases once more, recursively calling Examine-node on NP3. It is now three invocations deep. The dotted line shows its path.



spoken: *"For any thing."*

After processing np3, the controller will leave the np and thepp, go to the next constituent of **clause1**, use the dictionary entry for implications, and so on, et cettera.

The design of this generation component is oriented around the decision making process of the dictionary entries (see [McDonald 1978b] for more discussion). The principle reason that the process is deterministic and indelible, for example, is to simplify the conditions that the entries will have to test for. A more relevant example here is the use the controller to "pre-calculate" certain relations about the context and make them available through the values of recursive state variables maintained by Examine-node For example, the controller keeps pointers to the "current-main-clause", "current-verb-phrase", etc.. It keeps track of whether it is in a subordinate context, of what the last constituent was, last sentence, and so on.

Any of those relations could be calculated independantly by directly examining the form of the constituent tree and the annotations on its nodes and embedded message elements. But the point is more than just efficiency. By making certain relations readily available and not others, one says that just those relations are the important ones for making linguistic decisions A one of a kind operation like subject-verb agreement will have a special predicate written for it that "knows" where to find the relevant subject constituent in the constituent tree. But relations that are often used, particularly those needed for evaluating pronominalization, are maintained by the controller, and, as a corollary, are only available in their pre-computed form when the controller is present at that point in the tree.

The design of the controller guarentees that the generation proces will have these properties: (1) It is done in one pass - the controller never backs up. (2) Therefore decisions, choices of phrasing, must be made correctly the first time. (3) It is incremental. When the first part of the text is being printed out, later parts will be in their internal form. (4) Therefore very specific facts about the linguistic characteristics of earlier parts of the text are available to influence the decisions made about the later parts. (5) In particular, when the time comes to render any particular

was-a-thing, vs. **was-a-proposition** once and for all and makes it unnecessary for the heuristics that refer to this distinction to repeatedly include all of the particular cases. For that matter, it is also unnecessary to rewrite the code for the heuristics every time there is a new definition for a feature.

Other syntactic features currently computed include measures of relative position like **same-simplex**, **same-sentence**, or **stale**, and **proceed-and-command**, whihc are computed from the several position indexes in the record. The record of what constituent slot the last instance was in, in conjunction with the clause indexes, is used to check for features such as whether the last instance was the **previous-subject**. Also, parallel positions within conjoined phrases are noted.

Once the list of features is computed, the heuristics are run. At the moment, they are implemented as simple conditionals. Here again, there can be an immediate yes or no decision, or else a yet more involved process is invoked (see below). The grammar forces an immediate decision when **proceed-and-command** applies. Otherwise, a number of heuristics will immediately cause a pronoun to be used if there are no "distracting" references to other object in that vicinity of the discourse. For example, if the last instance of the object was itself realized as a pronoun, this will cause an immediately decision to use one again.

In the case of this example, the third instance of "x" will be described as:

same-sentence, last-subject, was-a-thing

As there are no other similar references in the vicinity to distract the audience, the heuristics will immediately decide that a pronoun should be used. The subroutine for computing the correct print name for pronouns is then consulted, and the result, "*it*" is returned to be inserted in the constituent tree and "spoken" on the next loop of the controller.

Reasoning about distracting references

Except when instance and anaphor are in the same simplex clause, syntactic relations alone are never enough to dictate whether or not a message element should be pronominalized. The linguistics component must to be able to tell if there are any other elements with which this one might possibly be confused. The problem is, of course, that the "confusion" will be a semantic or pragmatic one, i.e. it will be based on cognitive facts about the message elements which the linguistics component, *per se*, knows nothing about.

Given an oracle to tell it which message elements would compete with current one for the interpretation of a pronoun in that position, the linguistics component can use a simple procedure to decide whether to go ahead with the pronoun, namely to run those other elements through the pronominalization heuristics as well and see which accumulates the best reasons for being pronominalized.

Consider this example sentence. Imagine that the linguistics component has reached the point in brackets and must make the choice whether to say "*her*" or "*Candy's*".

"*Candy asked Carol to reschedule {her, Candy's} meeting for earlier in the day.*"

Whether or not two objects will be ambiguous depends on what the audience knows. In this case, an audience that knows who both Candy and Carol are will know that Candy is a graduate student who might well organize a meeting and that Carol is a group secretary, someone who would probably make the arrangements needed for changing a meeting's time. For such an audience, it would be not at all confusing to say "*her meeting*". An audience that didn't know who they were however would at best be confused and would in fact probably make the wrong choice.

This kind of information is much too specific to imagine encoding as part of general purpose dictionary entries. But because of the general unpredictability at the message level of whether an object will have subsequent references made to it in the eventual text, the linguistics component will have to make its query to the main program "oracle" at the very last minute as part of pronominalization heuristics.

The oracle will presumably be some kind of audience model. But for present purposes, we can think of it as a function that takes the object we are interested in ("Candy") as its argument and returns a list of those objects that appeared in this and recent messages which the audience might confuse with it. So, in this case, if the audience knew Candy and Carol, then the oracle would return a null list, and the pronominalization option would go through. If they didn't know them, then it would return "(Carol)", and a further round of heuristics would be tried.

To compare the relative "pronominalizability" of several message elements, Pronoun? runs them separately through the analysis and evaluation procedure. But instead of acting on the evaluation directly, it makes a list of the names of the individual heuristics that each passes and then compares the two lists. In the current program these would be:

```
Candy
    same-sentence
    proceed-and-command
Carol
    same-simplex                 ;via a trace
    proceed-and-command
    upstairs-subject
    no-interveening-distraction
```

In this case, the relative number of heuristics alone would indicate that Carol would make a "better" interpretation for a pronoun in that position, and that, therefore, the possibility of a using a pronoun for Candy should be rejected. But actually, the different heuristics are given weightings. *Same-simplex*, for example, is much better evidence than *same-sentence*.

Non-pronominal subsequent references

Every subsequent reference is first checked for the possibility of using a pronoun. If this check fails, a summary vector of the features analysed and of heuristics passed and failed is passed along to the message element's dictionary entry. Entries may have their own idiosyncratic procedures for dealing with these situations, but they may also make use of general procedures packaged by the grammar.

As explained in [McDonald 1978b], the "thinking" part of a dictionary entry consists of a set of "filters", which, if their conditions are met, will execute one or more "realization strategies" which assemble the phrase or modifer that the filter set decided upon. Because entries are not evaluated directly but instead are interpreted, it is possible for the interpreter to dynamically add or subtract filter sets according to the grammatical (or rhetorical - see below) circumstances.

One of the more common reasons for rejecting the use of a pronoun is that it might be missinterpreted as refering to some other object. The form of subsequent reference eventually choosen in these cases must distinguish the object from the one it is potentially ambiguous with, but does not have to recapitulate any more detail.

In particular, one frequent pattern for an initial reference is a noun phrase with the name of a class of objects as its head word, with a series of adjectives, classifiers, or qualifying phrases surounding it. There is a simple formula for constructing a non-pronominal, subsequent reference to follow this kind of NP namely to repeat the class name as the head word and use either "*that*" or "*the*" as a determiner.

Part of an element's discourse record is a list of the realization strategies that were used in the construction of previous phrases. This is a technique for smoothing over the irrelevant detail of the actual phrase that what used. As the realization strategies are refered to by name, can be annotated with properties describing what they do, and entered into abstraction hierarchies, Routines that have to think about what other routines have done or might do can do so at whatever level of generality is appropriate. In particular, this is a way to describe patterns of noun phrase construction so that general purpose filter sets can recognize them.

The initial references pattern above is recognized by a filter set that the entry interpreter can add. The filter's predicate checks for the name of the realization strategy **head<-classname** being included as one of the **"strategies-used"** of the anaphor. If it is found, this filter set will take precedence over any others in the entry. The filter set's action will assemble a new noun phrase with the same class name, as used for initial references (it is recorded with the entry), and either *the* or *that* as the determiner depending on a heuristic measure of the distance between this instance and the last. This is the process operating in a sentence like:

"There is room for a block on a surface iff that surface is a table or has a clear top."

Subsequent references to the same kind of object

The controller makes only one pass through constituent tree, turning internal, message level structures into linguistic structures as it passes. While the amount of information available for material behind the controller is limited only by how much annotation the designer cares to record, material in front of the controller is only megerly described. The (potential) linguistic properties of an object embedded in the constituent tree in front of the controller can be explored to a limited extent by "querying" its dictionary entry. However, this is limited as a practical matter because the interveening text has not been finished and any filters in that entry which depended on the discourse context will be undefined.

This means that if you want the realization of two separated objects to be coordinated, the coordination has to be planned for well in advance and somehow marked. Otherwise the first object will be realized freely, since it would not be able to "see" that there is even a second object present. The phrases below are examples of where coordination is required (The first two are from the tic-tac-toe talking program of [Davey 1974]. He used special purpose routines to handcraft the pairs.)

".. my edge and yours .."

"...a corner ...the opposite one "

". will enclose X's in square brackets and Y's in angle brackets"

"..a big block and a little one"

In each of these cases, the two objects were both of the same "sort": edges, corners, brackets, or blocks. By the usual criteria, this would mean that they share dictionary entries, and, indeed, the paired phrases have much in common, and could be seen as only differing in the choice of strategy for their adjectives and/or determiners. This means that the coordinating mark must be something other than the "kind-of" pointer that links objects with their entries. It will also probably have to be a temporary structure, since "*the opposite corner*" is a transient phenomena, defined only at particular moments in each game of tic-tac-toe.

The simplest way to mark the pairs is with an additional formula in the input message, e.g.

```
   (all-of-a-set corner1 corner9)
or (contrast-by-size B6 B3)
```

When the message is initially processed, formulas like these are indexed by their arguments so that, e.g., the dictionary entry for blocks will be able to notice them and choose its strategies accordingly.

Indicators like all-of-a-set are a part of the common grammar, and operate in the same way that the earlier filter set for subsequent references by classnames does. The dictionary entry interpreter keeps track of the arguments to the formula and when the last of them is being processed, it "interupts" and preempts the choice of determiner to insure that it is *the*, indicating that the speaker intends for the audience to appreciate that there is no other corner (or whatever) left. (This is a simplification.)

Rhetorical context

Rhetoric is the art of persuasion [Aristotle]. Stylistic variations in ordering, word choice, use of function words, elipsis, etc. are potentially rhetorical techniques, if the speaker program (or rather its designer) knows when their use would have a particular desired effect, i.e. when their use would make the text more persuasive.

The rhetorical context will typically be just an additional parameter to be noticed by the entires and grammatical routines. The dimension that it adds, however, greatly increases the fluency of the linguistic component's output. The only problem is that rhetorical phenomena have not been studied much at all - they have been sweep under the rug of stylistic variations".

Goals about how to express the message's content can be specified in the message. They will have their own dictionary entries and end up determining part of the rhetorical context that accompanies the syntactic context. (At this writing, the details of the structure of the rhetorical context are still being implemented. What follows is a sketch.) Consider:

All of the pronominalization heuristics mentioned earlier were based on syntactic relations. However, there are other relations governing the understanding and generation of texts, which have to do with their rhetorical" or "discourse" structure. In particular, each region of text will have a focus - loosely speaking the object or action that that text is "about" (see [Sidner 1978] for an elaboration).

Pronominalization of subsequent references to the focused object is almost always obligatory. (There can be exceptions if the last several references to the object were pronominalized, and the intention is to "refresh" the audience s memory.) In the example with "Candy" and Carol", if the previous part of the discourse had been saying things about Candy, then she would have been established as the focus of that sentence. Then the presence of a current-focus heuristic in Candy's list of sucessful heuristics would have outweighed all of the syntactically based heuristics in Carol's list and the pronoun would have been used.

The only question is how to mark and monitor focus or any other rhetorical indicator. It is not a natural or even consistantly definable part of a syntactic constituent structure. Therefore it will have to be "tacked on" somehow. The technique I am experimenting with is to implement a focus "register" which is explicitly set and reset by any dictionary entries that effect focus. A new message could also effect the focus register via an explicit directive included with it - say, when the topic of conversation is being changed. An explicitly dictated focus would cause the linguistics component to "transform" the realization of the content parts of the message to insure that the new focus is properly marked as such by the syntactic form of the text.

* * *

The rhetorical context could be very domain specific. Consider the sentence:

The black queen can now take a pawn."

Notice that it is not necessary to say "*a white pawn*" because immediate inference that one makes about what pieces it is legal for a piece of a given color to "take".

Since the criteria for constructing a refering expression for any chess piece will overlap, they will likely share a dictionary entry. Thus we have a sort of subsequent reference phenomena. The entry for chess pieces will be looking for the mention of a piece s color earlier in the text. If it finds one, or rather if it finds one of the complementary color, and if the situation is right, it can omit any mention of color from the phrase it has assembled.

How to determine that the situation is "right" is a matter for the rhetorical context to specify. The problem is the color of contrasting piece can be omitted only if the choice of verb or some other device indicates that, in fact, a constrasting context is present. But there are too many suitable verbs to imagine listing them in the entry and explicitly looking for them.

Instead, the rhetorical context will include a list of "relations" that currently hold. What relations there should be is a matter of the rhetorical roles that different parts of a message might play and whether the recognition of these roles by the audience could be facilitated by a choice of wording (i.e. it is a matter of research and experiment). For a program that talked about chess games, one of these relations would be:

```
opposing-pieces
    piece1 = xxx
    piece2 = xxx
    relation-name = {attack, defend, pin, ...}
```

To decide whether to include the name of a piece's color, the entry looks to see if there is an **opposing-pieces** relation holding at the moment. If there is, it looks to see if its piece is part of the relation and whether it is the second of the two to be mentioned. If so, it omits the color name.

The power of this representational technique is that it compiles its record of the needed facts at the time when they easily determined, i.e. as the message is being compiled, well before the relation name has been rendered into English and the simplicity of the relation obscured.

This technique should be applicable to many more phenomena than simply subsequent reference. Consider sentences like these:

"Brian also wants to come to the meeting."

"Mitch as a class then and so does Beth."

"The meeting might run overtime, but I don't expect it."

The underlined words are not a part of the "literal" content of those sentences. They represent rhetorical relations between parts of the sentence or between the sentence and earlier parts of the discourse.

If the source messages for those sentences described only their literal content, it would be impossible to motivate the use of *also*, *so*, or *but* in those ways, yet they are what give the sentences their naturalness. But if those rhetorical relations are included as part of the linguistic context, with their links to specific phrases and dictionary entries, including these "little" words becomes simple.

References

Aristotle, Rhetoric and Poetics, translated by Roberts, Modern Library edition, Random House, Newy York, 1954.

Davey [1974] **The Formalization o Discourse Production**, Ph.D. thesis, Edinburgh University.

McDonald [1978a] *"How MUMBLE translated the barber proof"* manuscript being readied for publication, MIT A.I. Lab. Cambridge, Mass.

McDonald [1978b] *"A Simultaniously Procedural and Declarative Representation and Its Use in Natural Language Generation"* in the proceedings of the 2d Annual Meeting of the CSCSI/SCEIO, Toronto, Canada, July 19-21, 1978.

Sidner [1978] *"The Use of Focus as a Tool for Disambiguation of Definate Noun Phrases*, this volume.