

# COREFERENCE IN A FRAME DATABASE\*

Steve Rosenberg  
Lawrence Berkeley Laboratory  
University of California  
Berkeley, CA 94707

Bruce Roberts  
Artificial Intelligence Laboratory  
Massachusetts Institute of Technology  
545 Technology Square  
Cambridge, MA 02139

This paper analyzes coreference within a frame-based semantic system. We discuss mechanisms for determining whether a new piece of knowledge (expressed as a frame) is actually a previously known referent. We propose frame database heuristics that allow new knowledge to be uniquely disambiguated for coreference so that concepts initially represented in many forms can be assimilated to those parts of the database that express the same knowledge, though perhaps in a different form. We hypothesize that these coreference mechanisms can be used by a text processing module whose control structure makes use of the organization of the text to reduce the combinatorial explosion of coreference searches in a database.

## 1. Introduction

We will present in this paper a computational model of coreference. We hypothesize that reference in text makes use of an underlying, modality-independent coreference module. When we understand a piece of text, we translate the individual sentences into some formal linguistic representation such as an annotated surface structure [2] with associated case information. This linguistic representation can in turn be translated into a "deep structure" consisting of (a set of) partially instantiated frames. These candidate frames will be the formal input to our reference mechanisms, whose output will consist of target frames located in the database, which are coreferential with the candidate frame. Thus frames represent the formal semantic objects on which our reference mechanisms operate. An example of transforming text into an internal frame representation is the PAL system [6,8]. PAL takes English input from the domain of office scheduling and maps this first into an annotated surface structure and from this into FRL. FRL will be briefly described in Section 2. It serves as our formalism for representing meaning as frames.

The translation from English language text into a formal semantic structure is independent of the coreference module, which operates on the frames and not on the text input. By viewing text processing in this light, we transform the problem of intersentence reference into the more tractable problem of how, given (a) some semantic representation, which we will formally specify in FRL, and (b) some new piece of knowledge, which can also be represented in FRL, we can determine if any frames in (a) are coreferential with those in (b). Thus reference in text

\*This report describes research done by both authors at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. It was supported in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-75-C-0643.

processing is a special case of the general problem of determining coreference in memory. FRL in fact provides a general formalism for representing knowledge gained from several sources, not just linguistic. Each mode of input may be structured so as to support reference mechanisms that are peculiar to the mode. Language, for example, contains a number of special reference mechanisms that make use of such particularly linguistic evidence as pronominal reference. However, since information once translated into FRL is modality independent, we propose that a modality independent coreference module supports these more specific reference mechanisms.

A coreference module is necessary in large databases where knowledge is constantly being accessed, added and removed. There is a danger that new knowledge will actually be identical to, or closely related to, what is already known, but that this relation will not be perceived. In fact, the size of the database is less critical than the range of allowable descriptions of the data. If Fahlman [3] is right, then implementing search mechanisms in a hierarchically organized database will be efficient regardless of its size. But even in a database containing only a few facts, one would like to be able to refer to them in a variety of ways.

## 2. Knowledge Representation

Roberts and Goldstein [7] have developed a working frame system that forms the basis for our knowledge representation. Their Frame Representation Language has been used successfully to represent knowledge from a variety of domains: office scheduling [4], the wheat commodities market [9], places and travel [5]. FRL extends the traditional characterization of properties as attribute/value pairs to allow comments (meta-knowledge), abstractions, defaults, constraints and attached procedures. A frame is a named collection of slots and forms the semantic definition of a concept. A slot has an arbitrary number of user- and system-defined facets. In addition to the Value facet, useful system defined facets are: Default, which specifies a default value;

Require, which specifies procedural constraints on the values for that slot; If-Needed, which specifies procedures that compute a value for the slot; and If-Added and If-Removed, which specify actions to be taken when a value is modified.

FRL allows concepts (represented as frames) to be arranged in an inheritance hierarchy. The value of the AKO slot (meaning "is a kind of") is a generic frame of which the current frame is a specialized INSTANCE. Thus the frame system forms a tree structure. Generic knowledge is stored higher up in the hierarchy and shared by frames lower down; specialized frames specify new distinguishing characteristics. Generic knowledge, including all attached procedures, is inherited automatically. For instance, in Fig. 1, the frame Appointment-72 is AKO Appointment. This means that the requirement on the Participants slot of the Appointment frame applies to the Appointment-72 frame, although it is not specified explicitly in the description of that frame. The requirement states that Appointment can have (by definition) only two Participants. It is a procedure that evaluates whether or not the restriction is satisfied. Appointment-72 also has values for the various slots that form the definition of an appointment.

```
MEETING
  AKO          -VALUE- MEETING
  CLASSIFICATION -VALUE- GENERIC
  INSTANCE     -VALUE- APPOINTMENT, PARTY, CLASS
  PARTICIPANTS  -REQUIRE- (?? (AKO -VALUE- PERSON))
  PLACE        -REQUIRE- (?? (AKO -VALUE- PLACE))
  TIME         -REQUIRE- (?? (AKO -VALUE- TIME))
  SELF        -MATCH- (MATCHSLOTS PARTICIPANTS PLACE TIME)

APPOINTMENT
  AKO          -VALUE- MEETING
  CLASSIFICATION -VALUE- GENERIC
  INSTANCE     -VALUE- APPOINTMENT-66, APPOINTMENT-72
  PARTICIPANTS  -REQUIRE- (= (NUMBER !PARTICIPANTS) 2)

APPOINTMENT-66
  AKO          -VALUE- APPOINTMENT
  CLASSIFICATION -VALUE- INDIVIDUAL
  PARTICIPANTS  -VALUE- MARY, JOHN
  PLACE        -VALUE- OFFICE-75      ;John's office
  TIME         -VALUE- 11:45AM

APPOINTMENT-72
  AKO          -VALUE- APPOINTMENT
  CLASSIFICATION -VALUE- INDIVIDUAL
  PARTICIPANTS  -VALUE- BILL, JOHN
  PLACE        -VALUE- OFFICE-75      ;John's office
  TIME         -VALUE- 1:30PM
```

Fig. 1 -- Examples from the Frame Database.

### 3. An example of reference

The following scenario depicts a hypothetical dialogue between a person and a scheduling program. *Italics* indicate the person's requests, the normal font indicates the scheduler's response.

(S1) *What is the schedule for this afternoon?*

(S2) There is a meeting at 3 in Fred's office.

(S3) *Who will be there?*

(S4) You, Bill, and the Marketing people.

(S5) *When is my first meeting today?*

(S6) An appointment at 11:45, with Mary.

(S7) *Change the place of that sales meeting to be my office.*

(S8) OK.

(S9) *What will be discussed?*

(S10) The report on the recent advertising error.

(S11) *What was the report's conclusion?*

(S12) The incident was not considered harmful.

We will ignore the sophisticated language processing this scenario implies, and focus on the forms of reference visible in this passage.

### 3.1 Name Reference

A concept can be directly referenced through its names. Some names used in this scenario were: Fred, Bill, office, and Marketing people. Names in text reference the frame if it already exists in the active data base, or else cause retrieval of the generic frame and the addition of an instance of it to the database for the scenario. Suppose, after reading a sentence with "Bill" as its topic, we then are given a sentence containing "Marketing V.P.". Presumably this also is another name for the same frame (if this is Bill's job title). There are other names that can be used as unique designants of the frame, whose use results in a reference to the "Bill" frame if a previous use of a name has added it to the database. Each frame must have an associated list of allowable names which directly access it.

Names often serve as a shorthand for other forms of reference. They summarize the results of more complex reference processes, eliminating needless duplication of effort when the same reference reoccurs. Many of these names are widely used in general discourse ("O.P.E.C.", for example). Other names, however, are unique only within a specified context. The name therefore must be contextually bound. For instance, there are other vice presidents of marketing. Ambiguities among global names can only be resolved by recourse to contextual information. Names sometimes contain descriptors that can be used to distinguish among several possible frames (e.g., "this month's report" distinguishes a particular Report frame). These modifiers often require linguistic solutions (e.g., "this" versus "that").

### 3.2 Contextual Reference

Contextual reference is the use of a descriptive term in a context where it functions as a reference. In the scenario, the use of the phrase "The incident" in (S12) is an unambiguous reference to the "advertising error" of (S10). In the context of the discourse, only the "advertising error" is an incident. Other contexts might result in the phrase having a different referent. Contextual reference involves the use of a general term in a context where its referent

can be uniquely disambiguated. Repeated use of a particular contextual referent in a text causes it to become a local name. After its initial introduction, "incident" would refer to the "advertising error" throughout the scenario. If "incident" itself had names, these would also function as local referents to "advertising error".

Contextual reference is reference by description, where the description is contextually bound. Consequently, in determining these referents we will need two things: a search space that defines the local context, and a description. We shall see that using a hierarchical semantic system allows us to generate both of these from the referent. In contextual reference, the search space can grow very large, since such referents may not lie in the direct heritage branch of their referents. In a descriptively rich system it is possible to describe a complicated concept, object, or event in many ways. For instance, fighting in the Middle East could be described as "an international incident", "a border clash", "a skirmish", and so on. Or, consider a "forecast". Almost anything can be a forecast, since all that is required is that it not yet have occurred. However, we do not wish to make everything inherit from the Forecast frame. There may be no unique semantic arrangement in terms of a tree hierarchy, such that all terms are exactly superior to just those terms they dominate. Achieving this would require a tangled hierarchy; one in which a frame could be AKO several other frames.

### 3.3 Generic Reference

Generic reference is the use of names that have a direct relation to the referent in the heritage structure. Two frames may be coreferential if the heritage path of one entirely contains the heritage path of the other. If one frame matches another frame's heritage path, but descends deeper into the frame tree, we have a possible generic reference. There exists along the AKO link a direct path in the frame tree between these two terms. A reference to a frame is also a potential reference to any frames which inherit from it. The reverse is equally true. A daughter frame can be a reference to superior frames in its heritage path. This follows from the fact that on the one hand, the more generic concept subsumes the more specific one, while on the other hand, the more specific concept contains all the semantic properties of the more generic concept. In (S6) of the scenario, appointment functions as a generic reference to the meeting of (S5). Fig. 2 shows the hierarchical organization of meetings, appointments and other activities in our scenario. Generic links can be found by determining if the heritage path for a new frame contains any other new instances, and then matching the reference to the candidate frames.

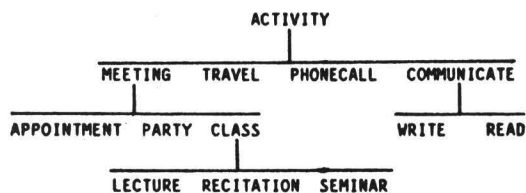


Fig. 2 -- A Hierarchy of Activities.

### 3.4 Frame Reference

Frame reference is a function of the empty slots of a frame. The sentences (S2) and (S3) can be linked through the realization that the meeting in (S2) has an unspecified slot for participants. (S3) specifies this slot in its request for information, eliciting the value of this slot in the reply (S4). Frame reference inherently involves forward expectations. Any new item in the database might potentially be a part of any empty slot in an existing frame. In a database of any size, checking all these empty slots for each new item is prohibitively expensive. Instead we think of empty slots generating expectations which search for an instantiation among new items. Thus frame reference involves the issue of control, as well as a matching problem. Later on in Section 5 we will discuss one mechanism, Sentinels, for controlling matching in FRL. Sentinals require a description of the referent, and this will involve the same matching processes as the other reference processes. They differ only in when the match is carried out.

## 4. Reference mechanisms in FRL

We will now refine our discussion of reference by developing a formalism within which we can extract the common elements and underlying mechanisms involved in these reference processes. Consider a general model for matching two frames. Matching a reference (candidate frame) to a referent (target frame) requires three things: (a) a search space; (b) a matcher that is capable of using (c) a candidate frame description that can be matched against frames in the search space to produce a set of target frames.

### 4.1 Specifying a search space

Consider first how two frames can be related within FRL. Any two frames will share a common heritage path in the frame tree descending from the topmost frame, the THING frame, which will branch at some daughter frame. For instance, in Fig. 2, both Appointment and Party inherit from Meeting. This branching node, together with the semantic features inherited from its father frames (such as Activity), defines the minimum semantic concept which includes both frames. It contains the intersection of semantic features common to any frame in one branch with any frame in the other branch of the heritage tree. Frames in the branching portions define semantic characteristics of the heritage path which are not common to both frames through additional slots on the frames themselves.

A search space for reference in FRL consists of a piece of the heritage tree which contains both referent and reference. We can specify this space by giving a node in this heritage tree which we know dominates both the referent and the reference. Call it a Location. Only frames below this node need be considered by the matcher.

The minimum search space is defined by the lowest node on the frame tree from which both frames inherit. If the reference and referent are on different branches, then the frame at which these

two branches diverge defines the minimum search space. Any frame lower down in the tree will exclude one or the other frame, while frames higher up will include more irrelevant frames. If the reference and referent are on the same branch, the superior frame defines the minimum search space. As we go up the heritage tree from the minimum search space, each new superior node will dominate more branches, until we arrive at the topmost node, the Thing frame. This defines the maximum search space; namely, the entire frame system.

We continue now with the overview of our model. Each reference is turned into a pattern to be used by the matcher. The pattern contains two kinds of information: a Location, specified as a frame name, that the matcher uses to define a search space; and a Description, specified as a pattern frame, that the matcher compares to frames in the search space.

## **4.2 Constructing the description**

A matching process comprises two parts: search and compare. The search space is defined by the Location, and searching is restricted to the branches descending from this node. A pattern is a frame-like description that can be compared against frames in the search space. It is a prototype of what the target frame looks like, containing information to delimit the search space as well as information about how to compare the frame with possible candidates. The pattern construction process differs slightly from one type of reference to another, but the pattern itself, once complete, is a unifying thread. It makes possible the use of a single general purpose matcher. The following two operations, common to all forms of reference, must be done to construct an initial partial pattern.

### **4.2.1 Collect requirements**

Each of the slots in the reference frame has a counterpart in the partial pattern. Defining characteristics for each slot of the pattern are gathered by inheriting the information in the slot's REQUIRE facet from each frame along the AKO branch of the reference frame. In addition, since FRL allows slot description frames that describe the role played by a particular slot in a frame, further requirements for individual slots of the pattern may be discovered by inspecting any slot description frame for that slot. Global requirements, applicable to the frame as a whole, reside on a special SELF slot of the pattern and are gathered from the reference frame just like the requirement on any other slot.

Requirements are either procedures to be applied to the value in the target frame or, in the case that the value itself is another frame, sub-patterns to be matched against the slots of the value. A typical requirement might check that the value is one of a set of permissible types, for example. Not all requirements are relevant to a particular match. Furthermore it is possible for specialized matching constraints to coexist in the same frame with requirements used for other purposes. Appropriate requirements are selected using identifying comments attached to each one.

### **4.2.2 Assign slot and frame matching procedures**

A corresponding process of collection and selection assigns a matching procedure to the frame pattern and its slots. These procedures occupy the MATCH facet of the reference frame, and will bear the local responsibility for determining whether the pattern matches a target frame. Again, the SELF slot supplies global knowledge; in this case, the procedure that controls the actual comparison between the pattern and a target frame. Explicit matching procedures for the frame provide the flexibility to vary the "weight" given to various slots and conditionally alter the course of the match. Matching knowledge is thus distributed throughout the frame hierarchy like other forms of procedural attachment in FRL.

Local matching procedures allow a pattern to specify for a particular match the actual criteria under which the match will succeed. Not all slots are relevant in effecting a match. Some non-essential slots may be safely ignored. The slot description frames already mentioned can supply other information useful to the matching process, since they include knowledge about the relation represented by the slot; for example, that it is transitive, or that it has an inverse? The kinds of matches that occur in practice will be discussed in a later section.

The details of the pattern construction process differ with the type of reference.

**Generic Reference:** Either a GENERIC or a INDIVIDUAL frame is specified, and this frame becomes the pattern with the addition of matching information to specify the direction of the match, from value to requirement (INDIVIDUAL), or from requirement to value (GENERIC).

**Frame Reference:** The matching procedures specify that the local and global requirements from only the unfilled slots are compared against the pattern.

**Contextual Reference:** The input is a prototype from the database. It has all the characteristics of a regular frame except that it is not a part of the AKO hierarchy.

**Name Reference:** The pattern is trivial -- a frame with the name used, or one looked up in the paraphrase list.

Since all referents are turned into a pattern, only one matcher is needed to handle all forms of reference. The different behavioral types of reference, which can be specified in FRL, can now be seen to really be underlying variations on a pattern containing two parts: a Location, specified in terms of a node on the frame tree; and a Description, specified as a frame pattern.

## **4.3 Matching frames**

Two frames may share all semantic features in common, expressed by identical slots, but contain different values for those slots. They are then different instances of a generic frame. We shall call this potential difference between two instances of a generic frame a pragmatic difference. Since procedural attachments are defined on the generic frame's slots, these will be held in common also. For example, the Appointment-72 and Appointment-66

frames are both instances of the generic Appointment frame. Both the Appointment-66 and Appointment-72 frames have a slot for participants, and inherit the same REQUIRE facet of this slot. However the value for this semantic feature will differ in the two instances; the two participants in the Appointment-66 frame are Mary and John, while the participants in the Appointment-72 frame are Bill and John. This pragmatic difference allows two instances to not be coreferential.

Matches between a reference and target frames in the search space will always have a semantic and pragmatic component. The semantic constraints can conflict. The two frames might not have identical semantic features (i.e. slots). It is not necessary for a successful match for both frames to have identical slots. In Generic Reference one set of slots subsumes the other. The inherent semantic relationship between generic referents means that we need only compare the common slots. We can assume the extra slots on one frame do not cause any semantic conflicts.

Contextual reference involves two frames that share some slots, but can each contain slots not on the other frame. It is possible that these differences exclude coreference. For instance, a Tree and a Dog each have unique slots when compared to each other, and are not coreferential. If a Dog has a slot for ambulation, whereas a Tree is explicitly marked as stationary, these two slots are mutually exclusive. (We exclude the case of conflicting slot values, as in a breathing slot marked "yes" for Dog, but "photosynthesis" for Tree.) By comparing the non-equivalent slots on different branches for mutual exclusion, the matcher can determine if frames on one branch are able in principle to function as contextual referents. However, it is also true that two frames whose slots form a partial set intersection can be coreferential. Take the case of a Father frame, which is AKO Person, and a Marketing V.P. frame which also is AKO Person. Clearly Fathers and Marketing V.P.s can each have properties that the other doesn't. Yet Fathers and Marketing V.P.s can be coreferential. Even a partial match requires that we are able to determine none of the slots on the Marketing V.P. frame conflict with those on the Father frame. In this case, the reference provides a best match, though not necessarily a complete match.

If there is no semantic conflict, the two frames are potentially coreferential. A match of the pragmatic features will determine this. (Note that by using the notion of a search space, very few inappropriate semantic matches need be entertained. We should seldom get in the position of considering whether dogs are references to trees, for example, since they do not bear a close relation in the frame tree.) This pragmatic match will compare values for the slots which are held in common. These values may be identical; they may differ, but be in slots which take a set of values; or they may conflict. If the slot accepts a set of values, and the values don't match, but meet the requirements of the slot, we may still be willing to consider it a match. Similarly, if the values conflict, but are ephemeral, like hair color, we may not require them to be identical unless there are multiple candidates to be discriminated among. One frame might have a slot value, while the other has none, but does have requirements; one value

may be a subset of the other value. If a slot on one frame has a value, but doesn't on the other, this value must be consistent with those that the value-less slot can take. New values for uninstantiated slots may trigger associated procedures that evaluate the potential new value.

The matcher makes use of procedural knowledge associated with frames since it does not itself contain the knowledge. Thus, (a) the matcher must know how to use this knowledge, which is not explicitly put in for matching (it is there for semantic definition), and (b) the matcher must make use of the fact that knowledge is distributed in the frame hierarchy and can be collected appropriately in the pattern. Not every slot needs its matching requirements explicitly stated if it can be inherited efficiently. The requirements for the particular slots will determine the outcome, and will also contain knowledge about how they can be relaxed, as in the case of values which change over time. Consequently there will be a degree of matching determined by the assigned value matching procedures. This allows us to formalize the notion of a good match as being a continuum.

## 5. Controlling the Matcher

The system we have been proposing is an "ideal" system in that it ignores the time dimension. Both reference and referent exist at the same time. However, information, as in the scenario, is understood over time. Reference may be forward or backward. A reference may be added at some future time, as well as already existing in the database. Indeed, there is no guarantee that the reference will ever occur! If for each new item in the database we always check to see if it is a reference, we will be making many useless searches. Therefore deciding when to compute reference is an important process. There is one class of reference where this choice can be reasonably determined. Frames will often have empty slots, and many of these slots are filled in by subsequent information.

To implement Frame Reference, we have chosen an approach that treats the empty slots of a frame as expectations. In practice, the decision to use forward expectations must take into account the extra computational burden this would place on the system. Maintaining forward expectations is useful to the extent that new information can be uniquely recognized by a set of special demons [1] called Sentinels created for the purpose. Sentinels were developed with James Stansfield by one of the authors (Steve Rosenberg). If all new frames are matched against all empty slots; a goal driven strategy of using expectations is equivalent to a data driven one of matching each new input against empty slots. However, by using the information available in the frame tree, we can reduce the number of matches each expectation requires, combining the best aspects of a goal driven and data driven approach in the notion of Sentinels.

Many demons in a database can slow processing, since the traditional notion of demons does not make use of the semantic organization inherent in frame based hierarchical system. Thus we introduce the concept of instance-driven demons. Consider a

demon attached to the instance slot of a particular frame; e.g., the Instance slot of Meeting. Whenever a new instance occurs that inherits from Meeting, the demon matches its pattern against that new frame. Thus, if the following sentence occurs:

(S13) *There is another meeting.*

another instance of Meeting is created, triggering this demon. Other new frames that do not inherit from Meeting (e.g., a new instance of Travel) do not trigger the demon. Such a demon will match against only a selected subset of new input. By choosing the appropriate frame to attach such demons to we insure that they match against only likely candidates with precise semantic relations to the frame.

Sentinels are initially goal-driven, in that they are created as active expectations. However, they draw on the information available about frames that can satisfy the expectation to place data driven triggers, called sensors, in appropriate places in the frame tree. These sensors are activated by only a small subset of the input, that subset with the highest likelihood of fulfilling the expectation. Whenever a sentinel is triggered, through the addition of a new instance that meets the sensor's requirements, it evaluates this triggering value, and determines if it fits the expectation. If so it adds the new instance as the value of the appropriate slot and frame. Once this is done the sentinel will erase itself and its sensors. Thus the system will not be cluttered with unneeded procedures that fire inappropriately. Sentinels are a way to create and manage demons in an organized semantic system. Two of their features are a limited life span, and the ability to move sensors around in response to a condition that is computationally determined as it develops. A sentinel can have several sensors which report to it. The sentinel is satisfied when some arbitrary logical or temporal conjunction of its sensors succeed. A sentinel has the capacity to evaluate conditional relations among its sensors, and even to remove current sensors and place new ones as a response to these conditional constraints.

In order for sentinels to efficiently encode expectations, they must be given a Location. New instance values inherit procedural attachments from up the tree. Thus a sensor on the Instance slot of the Thing frame would be triggered by any new instance added to the system. This is equivalent to testing every new instance against every expectation, exactly the case we wished to avoid by making use of the organization of the semantic network! On the other hand, if we do not know the precise form of the expectation, we would like to place the sensor at a location in the hierarchy where it is sure to be triggered by the right value, although there may be some false alarms. Thus the use of the frame hierarchy lets us place sensors where they will be triggered by semantic classes of items that are sure to contain the target items. If the description of the expectation is vague, we will need to accept more false alarms, but in any case we can use the organized hierarchy to make use of descriptions so that the expectations are optimally placed. In sparse databases such sentinels act as more efficient demon implementations. In rich databases, with much inferencing, they allow us to suspend processing at many places, since such processing will only be reactivated in highly appropriate circumstances.

Frame Reference uses the slot name and requirements to generate a pattern which defines a lowest possible node in the frame tree from which an instantiating value must inherit. For example, the Participants slot of a Sales-Meeting frame might inherit a requirement, in the form of a sub-pattern specification, from Meeting that says "AKO employee"; the slot itself has the additional requirement "includes Marketing-people". The sentinel on the instance slot of this indicated frame will examine each new token inheriting from this employee frame, and finally fulfill the expectation when at least two proper instances of employees have occurred, one of which also has a job slot value which is AKO Marketing-people.

## 6. Conclusions

We propose a model of coreference for frame databases. We discuss coreference from the behavioral standpoint in terms of the FRL formalism, and present a single elegant mechanism for handling all the coreference forms discussed. This mechanism consists of a pattern and a search space, both generated from the reference. The pattern is matched to items in the search space to generate target references. We briefly explore the issues involved in controlling such a reference module, principally through the use of Sentinels.

Once input has been translated into a frame representation, the coreference problem becomes independent of the medium in which the information was originally expressed. An intimate connection does exist between the form of the memory and the operation of the reference mechanism. We view text or discourse structure as providing a control mechanism which drives the underlying processes of memory. Thus the use of themes and topics, as well as more specifically linguistic clues, restricts the context for reference so that it works efficiently.

## References

- [1] Charniak, Eugene "Toward a model of Children's Story Comprehension". MIT AI-TR-266, December 1972.
- [2] Chomsky, Noam "Conditions on Rules of Grammar", *Linguistic Analysis*, 2:303.
- [3] Fahlman, Scott E. NETL: A System for Representing and Using Real-world Knowledge. MIT Press, Cambridge, MA, 1979.
- [4] Goldstein, Ira and Bruce Roberts "NUDGE, a Knowledge-based Scheduling Program". In *Proc. IJCAI-77*, Cambridge, MA, August 1977, pp. 257-263.
- [5] Jeffery, Mark J. "Representing 'Place' in a Frame System", M.S. Thesis, MIT, 1978.
- [6] Marcus, Mitch A Theory of Syntax Recognition in Natural Language. MIT Press, Cambridge, MA, 1979.
- [7] Roberts, Bruce and Ira Goldstein "The FRL Manual", MIT AI Memo 409, 1977.
- [8] Sidner, Candace "A Progress Report on the Discourse and Reference Components of PAL", MIT AI Memo 468, 1978.
- [9] Stanfield, James L. "COMEX: A Support System for a Commodities Expert", MIT AI Memo 423, 1977.