COMPUTING DISCOURSE CONCEPTUAL COHERENCE: A MEANS TO CONTEXTUAL REFERENCE RESOLUTION

BY

EZAT KARIMI, B.A.

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF ARTS

THE UNIVERSITY OF TEXAS AT AUSTIN

August, 1984

ACKNOWLEDGEMENTS

I am deeply indebted to my advisor Dr. R. F. Simmons, for his infinite patience, constant encouragement, and inspiring enthusiasm. His problem motivation and technical guidance were major influences in the shaping of this thesis. Without him, this work would not have been possible.

I would like to express my appreciation to Dr. C. Lengauer for his helpful criticism and suggestions of this thesis. Additionally, I would also like to thank Dr. M. Smith for his helpful comments.

Finally, I am grateful to my parents and brother for their consistent encouragement and love.

Ezat Karimi

The University of Texas at Austin
August, 1984

ABSTRACT

This thesis discusses the problem central to the interpretation of the discourse of a text: contextual reference resolution. The problem itself devolves to problems about the nature of the inferencing mechanism, knowledge base organization and discourse representation. A framework for an inferencing mechanism based on a theory of discourse coherence and focusing is discussed. A framework is described for a knowledge base which is composed of the knowledge about entities (through frame structures) and the knowledge about the coherence relations between different event/states. A model for discourse representation based an a small set of intersentential (coherence) relations and the relations between the conceptual structures for the entities discussed in the text is also presented.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Artificial Intelligence (AI), to some, is the science of teaching computers to do what people do easily, but computers don't. For this reason many AI scientists concentrate their work on studying the cognitive human behavior of natural language understanding in general, and text understanding in particular. Their studies show that the understanding process which appears so trivial and simple to the human reader, is really a series of complex functions which the human reader, (due to years of experience), no longer thinks of as a process.

To understand a text, the human reader has to resolve the ambiguities in the text, and recover the intended relation between what he is reading and what he already understood from the text. Understanding involves inferencing, in which the reader employs a complex chain of reasoning operating on his world knowledge (concepts) and his accumulated interpretation of the discourse of the text.

To construct a "natural language text understander," for a computer we are faced with problems at the design and implementation levels. We must deal with the difficulties of designing a systematic inferencing mechanism which can efficiently decide which part of world knowledge it needs to work with. At the implementation level we have the problem of defining a flexible data structure which can handle the dynamic nature of our knowledge base. Our goal in this research is to tackle these types of problems. Our approach is called **TEXT-ANALYZER**.

TEXT-ANALYZER is based on the notion of discourse cohesion recognition. The idea is that to dynamically build up a coherent discourse structure for a text, a system has to recover the coherence (logical) relation between each incoming sentence[1] and a sentence in the preceding discourse. The process of finding such a relation involves a chain of inferencing which would result in resolving the ambiguities in the current sentence. Consider example 1.1.

```
1.1.a Kim needed some money.
    b He went to the bank.
```

To determine what "he" and "bank" are referring to, and what the second sentence implies regarding the previous part of the text, the following inferences may be made:

1. usually when one (for example Kim) needs "money" he tries to get it,
2. One way that "Kim" can get some "money" is to withdraw it from the "bank,"
3. to withdraw "money" from the "bank" one (say, Kim) has to "go" there.

therefore "he" must be "Kim", and "bank" is related to "money."

To build up a system based on the notion of discourse cohesion recognition, it must be supplied with:

1. knowledge about the entities discussed as well as the coherence relations among them,
2. an inferencing mechanism which can reason through the world knowledge to compute coherence relations and resolve ambiguities.

MAINSTR is the knowledge base of TEXT-ANALYZER, and **CONNECTIVITY** is its inferencing mechanism.

[1] We consider a sentence as a unit of a text that can not be broken into any smaller and still meaningful units.

MAINSTR consists of two major parts, first, **MAINI** which contains frame structures representing knowledge about each entity. Figure 1.1 shows a frame structure for "bank" in example 1.1.

```
bank1
  isa(bank),
  with([[employee,clerk],
        [customer,customer],
        [component,money]]),
  actions([[deposit,[agt,customer],[obj,money],
                                   [loc,bank1]],
           [withdraw,[agt,customer],[obj,money],
                                    [loc,bank1]]).
```

Figure 1-1: An Instantiated Bank Frame

The other part of MAINSTR is **MAINR**. MAINR contains the knowledge about coherence relations between different events and states. MAINR implements seven types of coherent relations which are taken from [Alterman, 1982]. The system also realizes the *detail* relation which is implemented differently from the other seven relations. The seven coherence relations which are fully defined in [Alterman, 1982] are illustrated through the following examples:

1. **ANTECEDENT** of "giving" something to somebody is "having" that thing; one "must" have something "before" s/he can give it to somebody.
2. **PRECEDENT** of "eating" is "being hungry;" if somebody eats something he is "probably" hungry.
3. An object "falls" as **CONSEQUENCE** of being "dropped;" if an object is dropped it will "certainly" fall.

4. ▪Singing▪ and ▪playing▪ happen in **COORDINATION**; one sings a song ▪while▪ one plays that song.

5. ▪Going▪ to a place is a **SUBCLASS** of ▪moving▪ toward that place; going to a place ▪implies▪ moving toward that place[2].

6. ▪Chewing▪ and ▪swallowing▪ are **SUBSEQUENCES** of ▪eating;▪ ▪in order▪ to eat something one must ▪first▪ chew it and ▪then▪ swallow it.

7. **SEQUEL** of ▪hitting▪ an object is ▪breaking▪ that object; an object would ▪probably▪ break ▪after▪ being hit.

Attached to each of these relations is a set of constraints to prevent the system from falsely connecting two event/states together. The set of constraints corresponds to the set of assumptions that should be made about the nature of some particular arguments in two sentences in order that a coherence relation can hold between them. For example, ▪having an object1▪ is the antecedent of ▪giving object2 to someone▪, only if object1 is the same as object2.

Procedure **DETAIL** computes the relation ▪detail▪ between two sentences. The detail relation exists between two sentences when one of them is describing an item in the other sentence. In example 1.2 the second sentence is describing the item ▪hotel▪ in the first sentence.

```
1.2.a John entered the hotel.
    b It was a 24 story building.
```

CONNECTIVITY's task is to build up a coherent structure for the discourse of a text. It performs its task at two levels: the sentence level and the text level (these two levels can be interleaved). At the sentence level, CONNECTIVITY instantiates the frame structures for each lexical item. At the text level, CONNECTIVITY tries to find a fit for the partially interpreted sentence in the already interpreted part of the discourse[3]. To do this,

[2] E1 is a subclass of E2 if E1 implies E2 but E1 is more specific than E2; not every ▪moving▪ is ▪going▪ but any ▪going▪ implies a ▪moving.▪

[3] Our notion of fit is consistent with that of Lockman and klappholz [Lockman 83]

CONNECTIVITY tries to recover the coherence relation between the sentence at hand and a sentence in the already interpreted part of the text.

TEXT-ANALYZER represents the structure of the discourse of a text, through a conceptually-based (rooted) interrelation graph of the concepts assigned to the contents of the text. The root of the graph corresponds to what is understood as a whole from the text. TEXT-ANALYZER would interpret the example 1.3 as a graph rooted in the concept of "eating jam." This graph forms during the process of connecting "jam" to "fridge" through concept "food," and connecting concepts "wanting jam" to going to "fridge" through "having jam" and "eating jam."

```
1.3.a Sue wanted some jam.
    b She went to the fridge.
```

1.1 Outline of the Remainder

In chapter two, we will discuss the organization of MAINSTR. We will also give an overview of how the system works.

Chapter three describes major issues regarding the interpretation of a text in general and contextual reference resolution in particular. In this chapter we will argue that an inferencing method based on the notion of discourse cohesion is the most reliable one. We will also talk about the notions of "**expandability**" and "**minimal connectivity requirement**". These two are used by the system to,

1. prevent the inferencing mechanism from connecting two sentences for which a relation is not apparent,

2. facilitate the process of connecting two related sentences.

At the end of chapter three, the results of experimenting with TEXT-ANALYZER on three different texts will be discussed.

Chapter four relates our work to work done by other researchers in this area. We will discuss particularly the approaches that consider the effects of discourse structure on the interpretation of a text. We classify these approaches as:

1. Those which use a pre-defined discourse structure as the basis for interpreting a text. As examples of this class of approaches we will talk about the work done by Grosz [Grosz 78], the Yale group [Schank 79] and Mike Smith [Smith 81].
2. Those approaches in which the process of interpretation of a text is the process of constructing the discourse for it. From this class, we will talk about work of Alterman [Alterman 82], Hobbs [Hobbs 78], and Lockman and Klappholz [Lockman 83].

In chapter five we review our findings, and discuss directions for future research.

Chapter 2

MAINSTR: A Knowledge Base

2.1 Introduction

A typical text usually contains some apparent ambiguities, mainly because the writer:

1. avoids reintroducing an item which he has already talked about; instead he uses anaphoric expressions to signal to the reader that he is still talking about the same thing. In example 2.1 "that" refers to "afternoon."

   2.1.a I like afternoons.
   b That is the best time to relax.

2. omits explicitly mentioning the relation between a newly introduced item and an old one. The relation between "book" and "book store" in example 2.2 is an example of such a case.

   2.2.a Jim needed some books for his art course.
   b He went to the book store.

3. omits explicitly mentioning what a sentence as a whole refers to; what intention the writer wants to communicate in writing such a sentence. In the example 2.2, Jim's decision to buy books is omitted from the text.

The process that the reader uses in order to recover the relationships between the items in a sentence and the sentence itself, with the rest of the text is called *contextual reference resolution*. Reference resolution involves inferencing; the process through which the reader reasons through his world knowledge considering what he has learned from the text so far in order to resolve contextual references. In chapter 3, we will argue that the most secure framework[1] for an *inferencing mechanism* is the one based on the notion of discourse cohesion recognition. The idea is that for a text to be meaningful, each incoming sentence must bear a logical (*coherence*) relation with a sentence in the preceding discourse. Recovering that relation requires a chain of inferences through which the contextual references are resolved. Consider example 2.3.

```
2.3.a Sally went to the hospital.
    b She was sick.
```

Here, the second sentence is a precondition of the first one. That is if "Sally" "goes" to a "hospital" she is probably "sick.". To derive the coherence relation in this vein, the system has to relate "she" to "Sally," and "sick" to "hospital."

Our system, TEXT-ANALYZER is based on this framework. In chapter 3, we will talk in detail about the basis of the system. In this chapter we discuss the organization of *MAINSTR*, the system's knowledge base. To justify the organization that we considered for MAINSTR, we need first to explain what an inferencing mechanism needs to know in order to be able to recover the coherence relation between two sentences.

2.1.1 Discourse Cohesion

In a typical text, an incoming sentence Si can be related to a sentence Sj in the preceding discourse if Si is an expansion of Sj. Si is an "*expansion*" of Sj if Si is relevant to what is being said in Sj. The relevance can be understood by two means:

1. sentence Si is to provide further descriptive information about some

[1] by secure we mean efficiency and reliability; the inferencer must be able efficiently to trace the correct relation between items.

items in the previous sentence. For example in 2.4, the second sentence is about the ■weight■ of ■rocket.■

2.4.a A V2 rocket stood in the New Mexico desert.
b It weighed 8000 pounds.

2. sentence Si is expanding the entire concept described by Sj. In this case, the relation between the event/states described by Sj and Si is either *taxonomic* or *temporal*. The text 2.5 is an example of the first case. The relation between ■shopping■ and ■buying■ is taxonomic; One of the subsequences of ■shopping■ is ■buying.■ In the example 2.6, there is a temporal relation between ■being sick■ and ■going to the hospital;■ a sequel of ■being sick■ is ■going to the hospital.■

2.5.a Sally went shopping.
b She bought two pairs of shoes.

2.6.a Sally was sick.
b She went to the hospital.

The logical relation between two sentences, one of which is an expansion of the other, is called the coherence relation between the sentences. Coherence relations are derived from the relationships between certain arguments of two sentences. In example 2.4, ■rocket■ must have ■weight■. In the example 2.5, ■she■ must be referring to ■Sally,■ and ■hospital■ must be a place that a ■sick■ person usually ■goes.■ In other words, the coherence relation between two sentences can be defined in terms of the relationships that should hold between certain arguments of two sentences.

An inferencing mechanism needs to have access to two types of information in order to be able to recover the coherence relation between two sentences. First, it has to have access to facts about entities. Second, it needs to have knowledge about the coherence relations; it must know under what circumstances each such relation holds.

MAINSTR provides this sort of information for TEXT-ANALYZER.

2.2 MAINSTR

MAINSTR consists of two major parts:

1. *MAINI* which contains the information about the entities of the world referred to by words,
2. *MAINR* which contains the information about coherence relations between different event/states.

The components of MAINI are shown in appendix A. Appendix B includes the information encoded in MAINR.

2.2.1 MAINI

Each item in a text is identified using two types of information: first, what a reader knows about the item prior to reading a text, second, what he learns about the item in course of reading the text. A typical reader, prior to reading a text, knows that ▪John▪ is a person, a ▪restaurant▪ has customers and employees, and it is a place that somebody goes to eat. But considering ▪John▪ as a customer of the ▪restaurant▪ is something that the reader learns after reading a text such as the one in the example 2.7.

```
2.7.a John went to a restaurant.
    b The waiter seated him.
```

```
2.8.a John has only one auto.
    b He owns a Ford.
    c John's auto is red.
    d His auto is a convertible.
```

There is no scientific theory that states how a human reader organizes the information that he compiles about an entity in his memory, but there are at least two ways that a computer program can store its knowledge of an entity:

1. asserting separately the different facts about that item in its knowledgc base. ▪John's Ford▪ in example 2.8 (from [Grosz 78]) can be identified through the following assertions:

```
a. of(john,auto)
b. of(john,ford)
c. color(auto,red)
d. type(auto,convertible)
e. isa(ford,auto)
f. isa(X,Y),color(Y,Z) ----> color(X,Z)
g. isa(X,Y),type(Y,Z) ----> type(X,Z)
```

2. assigning a unique frame structure to that entity and saving whatever it knows or learns about it under that structure. "John's Ford", in this approach, can be identified by:

```
isa(ford,auto)

ford([[isa,auto],
      [color,red],
      [of,john],
      [type,convertible]])
```

We followed the second approach because in many cases, it is more efficient than the first one. Efficiency[2]is gained because of:

1. economy in computation; reasoning about (questioning) a certain property of an entity is more economical through a single structure than trying to chain the scattered facts to reach the same solution. To find out that "the color of John's Ford is red" the axioms **a**, **b**, **c**, **e** and **f** must be used. While to make the same inference with the unique structure one can only use the first axiom and scan through the structure for "auto."

2. economy of space; usually, saving information about a certain entity through a unique structure is more economical than saving the same information scattered around the knowledge base. Compare the necessary space for saving the information obtained about "John's auto" in text 2.8 in each approach.

[2] Grosz, [Grosz 78] justifies consolidating several D-structures into one for the same reason. A D-structure is similar to the structural specification that we assign to an entity.

2.3 Organization of MAINI

Units of information in MAINI are procedures which upon execution return the frames for the conceptual structures of the entities that the frames represent. For example, procedure "person" returns the "person" frame if it is called with an input argument of "John."

To organize the information in MAINI in terms of frames we considered the following formalism:

1. MAINI should only contain the base frames for classes of entities. The conceptual structure for other entities references these basic frames. For example having a single frame "person" is sufficient to compute the frame structure for any entity which is a person, for example John, Mary.
2. each base frame should carry as little information as possible, simply that which differentiates the entity that the frame represents from other entities. The shared properties should be separately saved in a way that all the entities sharing those properties can access them. For example entities "bedroom" and "kitchen" have the properties of a "room" in common. It is not economical to store the information about a room in the structure of both of them. The frame structure for "kitchen" and "bedroom" can save the properties specific to "kitchen" and "bedroom," and the shared properties can be saved under the frame for "room." The information about being a room can transfer to the conceptual structure of "kitchen" (or bedroom) when necessary.
3. a frame structure should not carry information about the entities that it has some relation to. For example the frame structure for "school" would include the information associated with the item "school," and a link to "teacher" but it does not include the information associated with "teacher."

Organization of the information in MAINI corresponds to a rooted graph. The nodes are the frame structures for the entities. A link originated at a node I can be any of the following types:

1. inheritance link (ISA link) which means that node I inherits the information in the parent node,
2. the links which show the associativity between the entities. For example, the node "teacher" is connected to node "school" through the link employee.

Figure 2.1 shows the upper part of the graph. Figure 2.2 shows a lower part of the graph.

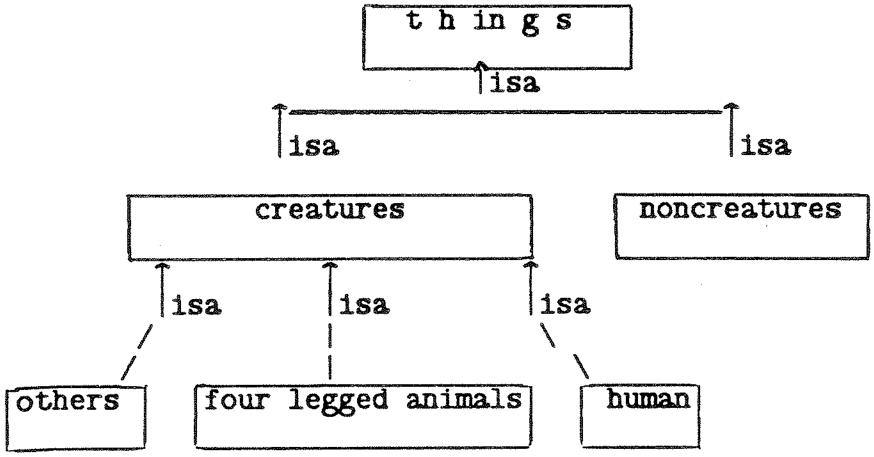

Figure 2-1: The Upper Part of the MAINI Graph

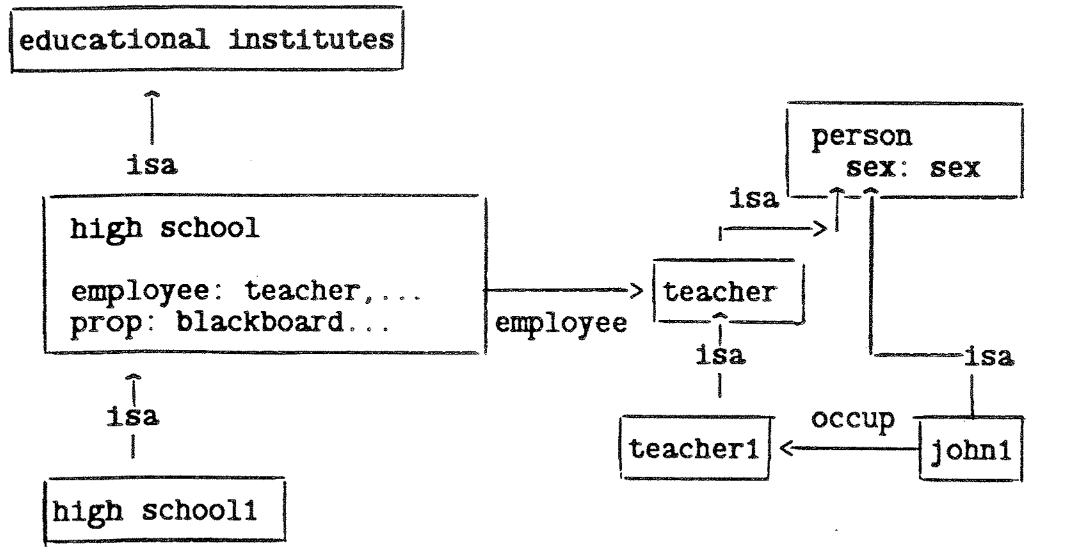

Figure 2-2: A Lower Part of MAINI Graph

2.3.1 Procedure for Instantiating a Frame

A frame structure in MAINI is designed to provide the basic information necessary for identifying an item that appeared in the text -- the basic knowledge that one (say, a computer program) would have about an entity prior to processing of the text. Specifically, a frame presents the conceptual structure of the item that the frame represents. This structure will be filled with the acquired information about the item during the process of interpreting the text. Figure 2.3 shows the instantiated frame structure for "supermarket." A frame structure has the following format[3]:

```
frame(Y,isa(X),
        with(Z),
        actions([Ev/St,Args]])))
```

[3] This format is similar to the one used by Smith [Smith 81] for frame representation.

```
supermarket1
   isa(supermarket),
   with([
        [employee,cashier],
        [customer,customer],
        [component,grocery],
        [prop,money]]),
   actions([[shop,[agt,customer],
                  [obj,grocery],
                  [instr,money],
                  [loc,supermarket1]]])
```

Figure 2-3: Instantiated Frame for Supermarket

Y is an instance of the entity for which the frame is instantiated. For example, john1 is an instance of John which in turn is an instance of person, and supermarket1 is an instance of supermarket.

X in the *ISA* slot shows the base concept for the instantiated frame. For example "person" is the base concept for the frame "john1.’

In the *WITH* slot, Z is a set of tuples *[Li,Ci]* which show the relationships between the item and other entities. Li denotes the relation. Ci is a dummy value. The real value for Ci is to be introduced by the text. For example after processing the text in example 2.9, the tuple [customer,customer] in the frame for supermarket (Figure 2.3) will be replaced by [customer john1].

2.9.a John bought some ice cream from the supermarket.
b He paid $5.

The *ACTIONS* slot represents the set of event/states that are usually associated with an entity. For example the event usually associated with "supermarket" is "shopping." The system uses the information in this slot in the process of connecting two sentences. For example, consider the following example.

```
2.10.a John went to a supermarket.
     b He bought some ice cream.
```

The system reasons as follows:

1. ▪John goes to supermarket▪ to shop (from the shopping concept associated with supermarket) for something.
2. When somebody is ▪buying ice cream,▪ ▪he▪ is shopping for it.

By comparing 1, and 2 the system concludes that ▪he▪ must be ▪john.▪

The frame structure for the entities which share some properties with other entities is computed in two steps. First, the frame which includes the information that specifies the item is instantiated, then the shared properties are added to the frame structure by means of the procedure ▪addp.▪ Figure 2.4 shows the intantiated frame structure for ▪kitchen.▪ Figure 2.5 shows the original frame structure for ▪kitchen.▪ The frame structure for ▪room▪ is also shown in the figure. W corresponds to the properties that differentiate a ▪kitchen▪ from other entities. Z (Figures 2.4 and 2.5) is the result of combining the value of W with the properties of a ▪room.▪

```
kitchen1,
   isa(kitchen),
   with([[component,food,fridge,oven,
                    counter,sink,door,window]])),
   actions([[cook,[loc,kitchen1]],
            [eat,[loc,kitchen1]]]))
```

Figure 2-4: Instantiated Frame for kitchen

The knowledge that the system acquires about an entity (in course of processing of the text) is reflected in the entity's conceptual structure by two means:

1. the relationships (between the entity and other entities) which are

```
kitchen
      isa(room),
      with(Z)
      actions([[cook,[loc,kitchen1]],
               [eat,[loc,kitchen1]]])))
                                    :-
                                    kitchen(W),
                                    addp(room,W,Z).


kitchen(with([[component,food,fridge,oven,counter,sink]]))


room
    isa(room),
    with([
         [component,door,window]]),
    actions([]))
```

Figure 2-5: Frame Structures for Kitchen and Room

recovered through the interpretation of the text are present in the frame. For example, after interpreting the text 2.10, the system recovers the relation customer between ■John■ and ■supermarket■ which is already present in the ■supermarket■ frame.

2. the text itself introduces some new properties for the item that is not considered in its frame structure. For example, in the frame for ■John■ no relation as customer is considered. But ■John's being customer■ of a supermarket is derived by the system in course of processing of the text. This new property is added to the frame for ■john■ in the form of [customer,[of,supermarket]] to the **WITH** slot in the frame for ■John.■

2.4 Coherence Relations

A pivotal step in designing a system based on discourse cohesion recognition is to find a set of coherence relations which is both sufficient and computable. By sufficiency of such a set, we mean that the relations defined in it can cover a large domain of coherence in a text. By computability we mean that a system be able to define such relations in terms of some elegant and simple procedures. Several different sets of coherence relations have been suggested by the researchers in this area [Alterman 82], [Hobbs 78], [Halliday 76], [Lockman 78]. The drawback of most of these suggested sets is that they are either hard to compute or are not sufficient [Hirst 81].

As mentioned in section 2.1, an inferencer deals with two classes of coherence relations in a text. First, those which occur between two sentences, one of which is expanding over an item in the other. Second, those which exist between two sentences when one is expanding over the entire idea expressed in the other. To deal with the first class of coherence relation, we defined the procedure *DETAIL*.

Procedure DETAIL deals with the conceptual structures of the items introduced in two sentences. In two ways it may assume of two sentences Si and Si+1 that Si+1 is expanding over an item in Si:

1. if the event/state described in Si+1 is defining one of the slots in the conceptual structure of an item in Si. For example in the following text, the second sentence is in **detail** relation with the first sentence. This is the case because "weigh" defines the slot "weight" in the conceptual structure for "rocket", (Figure 2.6).

```
2.11.a  A V2 rocket stood in the New Mexico desert.
     b  It weighed 8 tons.
```

2. if a case argument in Si+1 defines one of the slots in the conceptual structure of an item in Si. Consider the example 2.12. The second sentence is also in detail relation with the first sentence because the "fuel" case in the second sentence matches the slot "fuel" in the conceptual structure for "rocket." The case format for the second sentence follows:

```
rocket1
  isa(rocket)
  with([[fuel,fuel],
       [weight,[ton,[qty,eight]]],
       [loc,[desert,[type,newmexico]]],
       [type,v2]])
  actions(
       [flight,[ae,rocket1]])
```

Figure 2-6: Instantiation of a Frame

```
[carry,[ae,it],[pu,[fuel,[com,[oxygen,
                             [msr,[ton,[qty,five]]]]]]]]
```

```
2.12.a  A V2 rocket stood in the New Mexico desert.
     b  For fuel, it carried five tons of oxygen.
```

Notice that the procedure DETAIL checks that the item described as the affected entity or the agent of the second sentence is compatible with the item which is expanded upon. For example, DETAIL checks to see if "it" is the appropriate pronoun for "rocket" in the text 2.12.

Procedure DETAIL only checks the detail relation between consecutive sentences[4]. For example the second and third sentences of the following text will be in detail relation with the first sentence.

```
2.13.a  A V2 rocket stood in New Mexico desert.
     b  It weighed 8 tons.
     c  For fuel, it carried 5 tons of oxygen.
```

To deal with the second class of coherence, We chose the seven *conceptual coherence relations* suggested by Alterman [Alterman 82]. "A conceptual coherence relation is a relation that identifies concepts which

[4] The descriptive sentences should follow the sentence containing the item which is being described

cohere* ; it is a relation that two concepts frequently have in respect to each other [Alterman, 82]. We chose this set of relations mainly because:

1. they were easy to compute. To us, this feature was very important because we needed a set of relations which could be easily defined in order to demonstrate the practical use of coherence relations in the interpretation of a text.
2. by applying them to different types of texts (either those which already had been tested by Alterman, or those we tested ourselves), it was shown that they can cover a variety of relationships between event/states which means they were capable of covering a large domain of coherence in a text.
3. Besides those features, the representation of the discourse in terms of these relations is easy to interpret.

The seven coherence relations can be classified as *taxonomic* and *temporal* relations. The relations **antecedent**, **precedent**, **consequence**, and **sequel** are temporal, while **subsequence**, **subclass**, and **coordination** are taxonomic. Figure 2.7 which is from [Alterman,1982] illustrates these relations[5].

```
            Event/State Coherence Relations
                 /                    \
                /                      \
           taxonomic               Temporal
             /   \                 /      \
            /     \               /        \
     Whole/Part   subc        before       after
     |         |               / \          / \
   subseq    coord        antec  prec  conseq  seq
```

Figure 2-7: Characterizing the Relations

The second part of MAINSTR, MAINR is to define this class of coherence relations between different event/states.

[5] the definitions of these relations are given in detail in [Alterman, 1982]

MAINR contains a set of event/states which are paired according to their conceptual coherence relations. For example, the following is a component of MAINR:

```
  prec([eat,[agt,X]],[hungry,[agt,X]])
```

which is equivalent to:

```
    eat [[agt,X]]
      prec
        hungry [[agt,X]]
```

TEXT-ANALYZER uses the information compiled in MAINR to establish a connection between two sentences. Suppose, the system decides to connect the sentences shown in example 2.14.

```
  2.14.a John was hungry.
       b He went to the kitchen.
```

MAINR contains the following relations:

```
 prec([eat,[agt,X]],[hungry,[agt,X]])
 antec([eat,[agt,X],[loc,Y]],[travel,[agt,X],[dest,Y]])
 subseq([travel,[agt,X],[dest,Y]],[move,[agt,X],[twd,Y]])
 subc([move,[agt,X],[twd,Y]],[go,[agt,X],[dest,Y]])
```

The system, by using the above relations, establishes the following path between the two sentences:

```
eat10 [[agt,john1],[loc,kitchen3]]
  prec
    hungry1 [[agt,john1]]
      antec
        travel10 [[agt,john1],[dest,kitchen3]]
          subseq
             move10 [[ae,john1],[twd,kitchen3]]
               subc
                 go1 [[agt,john1],[dest,kitchen3]]
```

which means:

1. *John* eats because he is *hungry,*
2. to eat in the *kitchen,* *john* has to travel to the *kitchen,*
3. as *john* travels to the *kitchen,* he moves toward it,
4. as *he* moves *he goes* to the *kitchen.*

The coherence relations between pairs of event/states are defined in terms of a set of procedures. Each such procedure defines a coherence relation in terms of a set of constraints that must hold between certain arguments of two sentences. These constraints are equivalent to a set of assumptions that need to be made in order that a coherence relation shall hold between the two sentences. The general format for such a procedure is as follows:

```
Rel([V1|X],[V2|Y],X1,Y1,C,F) :- set-of-constraints
```

V1 and V2 are the verbs of the two sentences. X1 and Y1 are arguments of V1 and V2 before checking the constraints. X and Y are corresponding values for X1 and Y1 after the constraints are checked. F is the set of frames computed through the interpretation of the text. C is a set of tuples of the form [Ri,Di] -- the suggested pairing of the items after checking the constraints. Each such tuple denotes that the item Ri of V2 is associated with the item Di of V1.

Constraints are of four types:

```
check0(match([[Case1,Item1],[Case2,Item2]...]),
                                arg(Z1,Z2),F)

check1(match([[Case1,Item1],[Case2,Item2]...]),
                                arg(Z),changes(C))

check2(match([[Case1,Item1],[Case2,Item2]...]),
                                arg(Z),changes(C))

check3(match([[Case11,Case21],[Case12,Case22]...]),
                                arg1(X1,X2),arg2(Y1,Y2),
                                changes(C))
```

CHECK0 checks the existence of a relation between Itemi and the item denoted by case Casei. For example CHECK0 would allow the connection of ▪traveling▪ to a ▪place▪ and ▪eating▪ in such a place if ▪food▪ has a relation with that ▪place,▪ (for example kitchen).

CHECK1 requires that the argument of the case Casei be equivalent to the item denoted by Itemj. For example,

```
subseq([eat|X],[order|Y]...) :-
                check1(match([[loc,restaurant]...])...)
```

means that:

a ▪subsequence▪ of eating in a place is ordering if that place is a ▪restaurant.▪

CHECK2 functions similarly to CHECK1. It is different from CHECK1 in the following manner. CHECK1 would fail if the sentence being checked does not contain the case denoted in CHECK1. CHECK2 would add the case to the sentence (as a default) if the sentence being checked does not contain such a case[6]. For example in the following,

```
coord([grow|X],[produce|Y],...) :-
                        check1(match([[ae,plant],...]),...)
```

CHECK1 would fail if the ▪affected entity▪ of ▪growing▪ is not mentioned. But in the following,

```
seq([like|X],[tip|Y],...) :-
              check2(match([[recp,waitperson],...]),...)
```

if the ▪recipient▪ of the ▪tipping▪ is unknown, CHECK2 would add [recep,waitperson] to the sentence as a default.

CHECK3 checks the compatibility of the arguments of the cases Caseij and Casekl.▪ For example,

[6] CHECK1 and CHECK2 act similarly if the sentence being checked contains the case denoted by the checkers.

```
prec([eat|X],[hungry|Y]...) :-
                    check3(match([[agt,agt]])...)
means:
```

a precondition of **Z** "eating" something is **W** "being hungry" if **Z** is equivalent to **W**; for example, **Z** can be "John" and **W** a "customer."

The relationship between two items is recovered by two means: either through the direct processing of the frame structures for such items, or through the procedure *AGREE*. For example to recover the relation between "food" and "restaurant" or "food" and "kitchen", CHECK0 looks through the frames for restaurant or kitchen. If one of these frames specifies any relation with "food" for example as component, then CHECK0 confirms the relation between "food" and that item.

Procedures CHECK1, CHECK2 and CHECK3 use AGREE to detect a relation between two items. AGREE confirms the relation between two items in the following cases:

1. If one of the items is a more exact description of the other one. For example AGREE confirms the relation between "waitperson" and "waiter" because every "waiter" is a "waitperson" but not every "waitperson" is a "waiter."

2. AGREE also confirms the relation between two items if one of them is a pronoun, and the pronoun is in agreement with the other item. For example AGREE considers "he" and "John" as compatible because "John" is a male and "he" is the pronoun used to denote a male person.

Upon confirming the relation between two entities AGREE returns the proper value that replaces both items. Between "customer" and "John" AGREE would return John. Also "John" is the value that AGREE returns after matching "he" with "John." Checking-procedures after, receiving the replacement value from AGREE, reflect the changes in the argument of the sentences which are involved in the connection process. These procedures also keep track of all changes which occurred in the entire process of constraint checking. These "changes" are to be propagated through the entire path (explained below) of which the processed nodes (the two current event/states that a connection between them is just being confirmed) are part.

2.4.1 The Process of Connecting two Sentences

EXPAND's task is to establish a connection between two sentences. To connect two sentences **A** and **B** together, EXPAND has to construct an inferencing path between them. The path between **A** and **B** is a set of intermediate nodes **E1,E2,...,En** which are instances of conceptual event/states from MAINR such that the following relation exists:

```
(A Ci E1), (E1 Cj E2),...,(En-1 Ck En), (En Cp B)
```

Ci is a coherence relation from the set of the coherence relations defined in MAINSTR. Assume, for example, the system decides to connect sentences expressed in example 2.15,

```
2.15.a The seed is distributed by the waves of ocean.
     b It grows on a distant shore.
```

According to **MAINR**:

```
coord(distribute,carry)
prec(sow,carry)
antec(grow,sow)
```

These pairs form the following path between ▪distributing▪ and ▪growing▪[7]:

```
distribute
   coord
      carry
         prec*
            sow
               antec*
                  grow
```

[7] an asterisk <*> in front a relation indicates the reverse of the relation.

To find a coherence relation between a node such as **Ei** in the inferencing path and an event/state in MAINSTR, EXPAND chooses a coherence relation and searches in MAINSTR to check if there is such a coherence relation between **Ei** and another event/state. If that is not the case, then EXPAND will try another coherence relation. If EXPAND fails to find any node after trying all the coherence relations, it backtracks for one node. That is it undoes the last node found in the path, and tries another coherence relation with the node from which the failed node had been traced.

In the above example, if EXPAND were not able to find node "grow" by trying all the coherence relations with node "sow", then it would undo the path between "carry" and "sow", and would try other coherence relations to find a new node to pair with "carry." In case of the existence of such a node, EXPAND has to satisfy the set of constraints attached to the relation. The relationships between the items of two sentences are recovered as side effects of the process of satisfying these constraints. For example, for "sowing" to be an antecedent of "growing," the following constraints must be satisfied:

```
antec([grow|X],[sow|Y],X1,Y1,F,C) :-
     check3(match([[ae,ae],[loc,loc]]),arg1(X1,X),
                                        arg2(Y1,Y),...),
     check1(match([[ae,seed]]),arg(Y2),...).
```

which means that the affected entities and the locations of two events should be the same, and in addition the affected entity of "sowing" must be "seed. If EXPAND succeeds in finding a new node of the path, then its task is to propagate all the "changes" resulting from satisfying the constraints associated with connecting the last two nodes as modification to the already established part of the path and to the frame structures for the items included in the path. "Changes" is a set of binding pairs **[Ri,Di]**. Each such pair asserts that the item **Ri** should replace item **Di**. For example a pair [customer,john] says that the value of customer is to be replaced with "John." EXPAND's task at this point is to reflect such a change in the arguments of the nodes involved in the path as well as in the frame structures for the items involved in that path. Consider the following example:

2.16.a Sally wanted some ice cream.

b She went to the supermarket.

Part of the path that the system constructs in order to connect the second sentence to the first is as follows:

```
shop10  [[agt,customer],[loc,supermarket2],
                        [obj,grocery]]
  subc
    buy10  [[agt,customer],[loc,supermarket2],
                           [obj,grocery]]
      antec
        travel10[[agt,customer],[dest,supermarket2]]
          subseq
            move10  [[ae,customer],[twd,supermarket2]]
              subc
                go1  [[agt,customer],[dest,supermarket2]]
```

Through the construction of this path the system recovers the relation between the "she" and "supermarket" as "customer."

Then, to connect the first sentence to this path, the system uses the following path:

```
buy10 [agt,Sally1],[loc,supermarket2],[obj,icecream4]]
  conseq
    have10  [[age,Sally1],[loc,supermarket2],[obj,icecream4]]
      prec
        want1  [[agt,Sally1],[obj,icecream4]]
```

Through constructing this path the system finds out that object "ice cream" is related to object "grocery" and agent "Sally" to agent "customer." At this point AGREE suggests that "ice cream" should replace "grocery," and "Sally" "customer." This suggested change is propagated through the first path. Meanwhile, the conceptual structure for "Sally" would record "Sally" as "customer" of the "supermarket," and the conceptual structure for "ice cream" would record it as a "component" of the "supermarket." The conceptual structure for "supermarket" would also have "Sally" as its "customer" and "ice cream" as one of its "components."

Chapter 3

The Basis of TEXT-ANALYZER

3.1 Contextual Reference Resolution

Contextual reference resolution refers to the process of determining the relation (if any) between items in the current sentence and those in previous ones. That is, to write a new sentence, a writer selects a previously mentioned item and expands it one way or another depending on the reader's intelligence to resolve the references. To fully interpret such a sentence, the reader must recover what those items refer to. Generally two terms *referent* and *antecedent* are used interchangeably to symbolize the referred item. But, strictly speaking, the referent is the conceptual structure of a real world entity that a reference refers to. The antecedent of such a reference is another item in the text which through some relation refers to the same conceptual structure. In other words, the reference and its antecedent are corefering to the same entity.

If item R is referring to entity E2, and item D is referring to entity E1, then D is a possible coreference of R in referring to entity E2 if:

1. R is a pronoun for E2, and E2 is identical to E1:

```
3.1.a Mary always tells us weird news.
    b But she never reveals the source.
```

   In example 3.1 "Mary" is referring to the conceptual structure of a person which is referred to by "she."

2. E2 (E1) is an instance (example) of E1 (E2):

3.2.a Dogs are friendly animals.
b Mary's dog accompanies her every where.

In example 3.2, "Mary's dog" is an example of a "dog" which in turn is an example of an "animal."

3. E2 (E1) is part of E1 (E2):

3.3.a Mary went to the bank.
b She withdrew some money.

In above example, "money" is a component of the "bank." In example 3.4 (from [Hobbs 78]) "combination" is a part of "safe."

3.4.a John knew the combination.
b He opened the safe.

It has been long realized that the tool for resolving contextual references is an inferencing mechanism which has access to a large amount of world knowledge about things and their relations to each other. Such an inferencing mechanism uses that knowledge to construct a chain of inferences to result in clarification of a reference. Consider example 3.5.

3.5.a Stella turned up the radio.
b A minute later, the next door neighbor came by, complaining about the noise.

To recover the relation between "radio" and "noise" in example 3.5, the inferencer uses the following information encoded in the knowledge base:

1. a radio is an appliance which produces sound.
2. sound is kind of noise.
3. turning up the radio produces more noise than there was before, probably to an extent which is noticeable.
4. too much noise usually bothers people who are around.
5. if a person is bothered with something he usually complains about that thing.

6. to complain about something to somebody, one contacts that person.

7. one way to contact somebody is to go to the place that person is.

8. a neighbor is one who lives near by.

There are some serious difficulties with a world-knowledge-based inferencing mechanism. How can such an inferencer determine which portion of the knowledge base is relevant to the process of resolving a reference, which set of inferences would result in resolving of a reference, and when can such an inferencer conclude that a reference is resolved? We explain the problem through some examples.

Consider example 3.6 which includes the text shown in example 3.5.

3.6.a Stella's dog was barking in the yard.
b She turned up the radio
c so that the dog's barking would not bother her.
d A minute later, the next door neighbor came by,
e He complained about the noise.
f He asked her to keep her dog quiet.

While interpreting sentence **e**, the inferencer has no way of knowing that "noise" is referring to "dog's barking," not to "radio." If it uses the same information that it used in example 3.6, it would falsely conclude that the "radio" is the antecedent of "noise." One solution to this problem is to have the inferencer try other items (such as the dog's barking) in search for a proper antecedent. But then what would be its criterion for making a decision over selecting an antecedent?

Now consider example 3.7. "Noise" is referring to "radio." But the inferencer, even after realizing this relation, has no way of knowing that it had found the right antecedent, and will continue trying other items in the text.

3.7.a Stella's dog was barking in the yard.
b She turned up the radio
c so that the dog's barking would not bother her.
d A minute later, the next door neighbor came by,
e He complained about the noise.
f He asked her to turn down her radio.

The criterion for designing a knowledge-based inferencing mechanism is that it have a sound method for controlling such an inferencer, a method which would give it the capability of knowing what to infer and when to stop. Although there has not been a general method for controlling the inferencing mechanism, there are some theories which provide some solutions for the inferencer control problem based on the effect of the discourse structure on text comprehension. Most of these are discussed below.

3.1.1 Effects of the Discourse Structure on References

The interpretation of the discourse of a text is a set of sentences which elaborate on the topic which the discourse is about. In other words, to write about the topic, a writer chooses a set of sentences which describe different aspects (related concepts or subtopics) of a topic in a way that they describe what he wants to say about that topic. That is, the writer picks a topic and writes a few sentences about it. Then he writes about a subtopic (which becomes the new topic). After writing a few sentences, he may go back to the old topic or write about a subtopic of the new topic. The structure of such a text is a tangled hierarchy similar to the one shown in Figure 3-1[1]. The nodes denote the items that each sentence of a text is about. The arcs[2] symbolize the connectivity of the discourse; an arc originating from some node I points to

[1] Throughout this thesis we interchangeably use the terms topic and focus to indicate the item which is the center of attention in a sentence or sentences, although both terms imply slightly different notions. Consider the following examples:

```
John ate the bread.
It was hot.

John decided to take a trip to Hawaii.
He went to the travel agent.
```

"Bread" is the focus of the first sentence and the topic of the whole text. In the second example,"trip" is the focus of the first sentence. The focus of the second sentence is either "he" or "travel agent." The topic of the text is "trip." The focus indicates the center of attention in a sentence; the focus is the topic of the sentence containing it and the sentences which are the expansion of that sentence.

[2] the arcs actually correspond to the relationships between the items denoted by the nodes as well as the intersentential relations between the sentences containing such items.

some node **J** implying some kind of relation between the items denoted by those nodes; the sentences containing such nodes bear some kind of relation in respect to each other. We say that the sentence containing node **I** expands to the sentences containing the nodes to which arcs from node **I** point[3]. Consider the text shown in example 3.8.

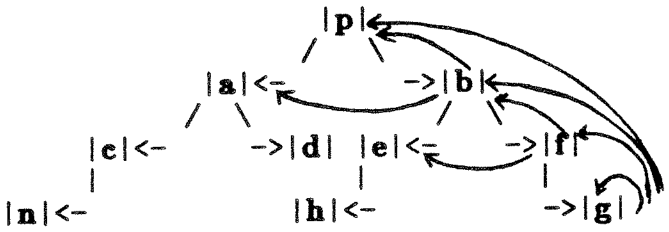

Figure 3-1: A Discourse Structure

3.8 a John decided to take a trip to Hawaii.
b He was very excited about it.
c He went to the travel agent.
d He bought two tickets; one for himself, one for Janet.
e Then he went home,
f to pack his stuff.

The topic of the text is "trip." The second sentence is an expansion of the first one -- an arc from trip to itself. The second sentence is also an expansion of the first one -- an arc from "trip" to "travel agent,"-- to take a trip one usually needs to get a ticket, and "travel agent" is a place to get it. Sentence **c** is expanded to sentence **d** -- an arc from "travel agent" to "ticket." It seems that sentence **e** does not result from the expansion of the so far encountered sentences. But sentence **f** is an expansion of the first sentence -- an arc from "trip" to "stuff"-- before one takes a trip he needs to "pack his stuff." Now again consider sentence **e**. It is connected to the rest of discourse through sentence **f**.

[3] Later in the chapter we will say that node I is considered as the item being expanded upon.

From what we said so far we conclude that each new sentence (of the text) is describing an item related to an item previously stated in the text. But if **S** is the list of all items previously mentioned in the text, then certain members of S can be chosen as the ones to be expanded upon (and the reader is still able to trace the connection). For example, the writer can not continue the above text with the following sentence:

It cost him 500 bucks.

which means the "tickets" "cost" "John" "500 bucks."

But he can continue with a sentence like:

It took him 2 hours.

which means the process of "packing" "took" "John" "two hours."

or he can continue with:

He planned for this vacation for a long time.

which returns to the original topic of the "trip."

A sublist of **S** (the list of items mentioned so far in the text) whose members can be expanded upon in an incoming sentence is called the **focus space**. Describing the focus space as a list implies that its elements are given different priorities. There is an important implication behind the relationship between an item and the elements of the focus space. If we define the antecedent of an item **g** as another item in the text that bears some relation with **g**, then such antecedent must be a member of the focus space for item **g**[4]

What was just concluded is tempting, because it implies that we could use such a list as a means for controlling the inferencer. That is, now the inferencer is only dealing with a limited list of items to search for an antecedent. Because of the priority given to the items in the focus space, the inferencer can select the first appropriate candidate from the focus space as the

[4] For more detail on the notion of focus space see [Grosz 78], [Sidner 83].

antecedent of a reference. But the problem is that the approach based on using the focus space as a source of antecedents is not always workable because for some sentences there is no precise way of computing the focus space without some knowledge about the nature of the sentence whose references are to be resolved [Lockman 83]. Consider examples 3.9, and 3.10. The focus of 3.9.a is "key" but the focus of 3.10.a is "John."

3.9.a John removed the key from the lock.
b It was not the right one.

3.10.a John removed the key from the lock.
b He entered.

3.1.2 Discourse-Cohesion-Based Inferencing Mechanism

Any new sentence in a text is either expanding (elaborating) on a topic expressed in the preceding part of the discourse, or introducing a new topic which has (will have) some relation to previously expressed topics. The writer follows some structural conventions in writing such a sentence to make it bear a certain (coherent) relation to the portion of the preceding discourse that it is supposed to expand upon (depending on what he wants to say). In turn, the reader interprets the sentence while he is trying to find a logical (coherent) relation between that sentence and an earlier part of the discourse (trying to figure out the intention of the writer for writing such a sentence). In other words, both the writer and reader continue (build up) the discourse such that it remains coherent. Such a discourse has the property of exhibiting structural relationships between its different portions depending on the propositional contents of the portions, [Hobbs 78]. In example 3.11, there is a cause relation between sentence **a** and **b**. That is, "Mary is tired" is a consequence of "her working."

3.11.a Mary is tired
b she was working all night long.

Coherently connected sentences have cohesive ties because they express related topics[5]. From what has just been stated we infer that two coherently connected sentences tend to refer to related items. Consider example 3.12.

3.12.a Stella turned up the radio.
b a minute later, the next door neighbor came by, complaining about the noise.

The two sentences are coherent because:

1. "turning up the radio" would cause too much "noise."
2. too much "noise" usually bothers other people.
3. as a consequence of being bothered by something, a person "complains" about that thing.

The cohesive ties between two sentences are "noise" and "radio," "Stella" and "next door neighbor."

Some researchers in this area have tried to take advantage of this close tie between the notions of coherence and coreference to come up with a method of controlling the inferencing mechanism. Control of the inferencer is achieved simply by having the inferencer determine either that there is a coherent relation between the current sentence and a portion of the previous discourse, or that no such relation exists. The inferencer will terminate after achieving that goal.

In this approach the process of resolving the references is part of the process of determining the coherence relation. Consider again example 3.12. The relation between "noise" and "radio" can be as follows:

[5] Note that the reverse of this definition is not necessarily true. That is, two sentences with cohesive ties are not necessarily coherent. Consider the following example.

Sally wanted some ice cream. She went to the bank.

"She" is related to "Sally" but two sentences are not coherent. For two sentences to be coherent, one should be contributing additional information to the other one [Hobbs 78].

```
1. turn up(radio) -cause--> too much(noise1)
2. too much(noise1)-sequel--> bother(P,noise1)
```

```
from sentence b, we infer:
```

```
3. complain(neighbor,noise2) <--sequel- bother(neighbor,noise2)
```

By comparing the right hand sides of 2 and 3, noise2 will be resolved to noise1.

In the remainder of this chapter we will discuss TEXT-ANALYZER, a system based on the discourse coherence recognition approach.

3.2 TEXT-ANALYZER

TEXT-ANALYZER consists of two major parts, its world knowledge base **MAINSTR**, and it's inferencing mechanism **CONNECTIVITY**.

The information encoded in MAINSTR are of two types:

1. a set of procedures which compute the frame structures for the real world entities which are referred to by the items in the text.
2. a set of procedures which compute the coherence relations between the event/states described in the sentences in terms of the chain of inferences that may be obtained from the propositional contents of those sentences.

CONNECTIVITY's task is to interpret each incoming sentence in relation with the preceding discourse while relying on the information encoded in MAINSTR. The interpretation process takes place in two steps. The first step is to produce a partial interpretation for the sentence at hand. This step is mainly composed of identifying the frame structures for the items present in the sentence and recovering their relationships as far as possible. Consider example 3.13. The case format for the sentence is also shown. As the result of the first step of the interpretation process, the frame "palm1" is created for palm. Notice that type and location of "palm1" is reflected in its frame structure (Figure 3-2).

3.13 A coconut palm appeared on a tropical island.

```
[appear,[ae,[palm,[type,coconut]]],
            [loc,[island,[type,tropical]]]],
```

```
palm1
   isa(plant)
   with([
         [seed,seed],
         [loc,island2],
         [type,coconut],
         [parts,branch,root,trunk],
         [height,height],
         [fruit,fruit]])
   actions([[grow,[ae,palm2],[loc,[]]]])
```

Figure 3-2: Instantiated Frame for the Palm

In its next step, CONNECTIVITY uses the information encoded in MAINSTR and the representation of the interpreted discourse to detect a connection that may obtain between an incoming sentence and a sentence in the representation. The contextual references are resolved as part of the process of detecting such a connection. If no such connection is possible (with any sentences in the representation) CONNECTIVITY assumes that the sentence at hand will have a coherence relation with a sentence yet to come which itself will connect with the preceding discourse.

CONNECTIVITY accounts for two types of connection between two sentences:

1. the second sentence provides descriptive information for some items in the previous sentence. Procedure **Detail** accomplishes the task of detecting such connection.
2. CONNECTIVITY is able to establish an inferencing path between two sentences. This path is equivalent to the set of inferences needed for recovering the coherence relation between two sentences. This path consists of a chain of coherently connected conceptual

event/states (either implicit or explicitly mentioned in the text) involving items included in two sentences.

To give a general view of how the system works, we describe the system process of interpreting the text shown in example 3.14.

```
3.14.a Sue went to the supermarket.
     b She bought some ice cream.
```

First, the frame structures for items "Sue," and "supermarket" are instantiated. Then, the system moves on to the second sentence and computes the frame structure for "ice cream." To find out that the sentence **b** is an "expansion" of sentence **a**, the system constructs the following path between the events described in those sentences.

```
buy1 [[agt,sue1],[loc,supermarket2],[obj,icecream1]],
 antec
   travel10 [[agt,sue1],[dest,supermarket2]],
    subseq
      move10 [[ae,sue1],[twd,supermarket2]],
       subc
         go [[agt,sue1],[dest,supermarket2]]
```

which means:

1. "To buy ice cream from the supermarket," "Sue" has to travel there.
2. "She" travels to the "supermarket" by moving toward it.
3. As "she" moves toward the "supermarket" "she goes there."

During the construction of this path, "she" is resolved to "Sue," "Sue" is related to "supermarket" as its customer, and "ice cream" is related to "supermarket" by a component relation.

3.2.1 Important Issues in Design of a Discourse-Cohesion-Based Inferencing Mechanism

To use an inferencing mechanism based on the notion of exploiting the local cohesion in the text, we are required to consider several issues [Hirst,1981]. These are:

1. what set of coherence relations to use,
2. how to detect these relations in a text,
3. how to use them to build a representation for the discourse structure,
4. how to search in this structure for referents

In chapter 2 we explained in detail the set of coherence relations that TEXT-ANALYZER uses. In the following sections we will discuss the remaining issues in the course of describing TEXT-ANALYZER.

3.2.2 A Representation for the Discourse

The discourse of a text is a partial ordering of some local contexts. Each such context corresponds to a set of coherently connected sentences which the writer uses to elaborate on a specific topic. These contexts are connected together through a set of sentences which relate the topics of the contexts together. Consider example 3.15.

```
3.15.a Sue wanted some ice cream.
     b She went to the bank.
     c She withdrew some money.
     d She went to the supermarket.
```

The first context is about "Sue's wanting ice cream" with "ice cream" as focus. The second context describes that how "Sue" acquires some money. The third context is about "Sue's going to the supermarket." The third context is connected to the second through the relation between "money" and "supermarket" which can be inferred as follows:

1. "one goes to the supermarket" to buy something,
2. in order to buy something, one has to have "money,"
3. one way to acquire "money" is to "withdraw" it from the "bank."

An enlarged context (resulting from connecting the third context to the second one) will be connected to the first one through the connection between "ice cream" and "supermarket" which results from the following set of inferences:

1. if one buys something then s/he has that thing,
2. a sequel of "wanting" something is to have that thing.

A TEXT-ANALYZER representation of each context is a graph of connected concepts which correspond to the contents of that context (it is called a schema[6]). Each node of the graph is either a sentence or the conceptual structure for an item. An arc in the graph is either the coherent relation between two sentences or the relation between an item and its conceptual structure. The relation between the objects is shown through the relation between their conceptual structures. Figure 3-3 shows the set of schemas (schemas a and b) created for the text in example 3.15 after three sentences of the text are processed.

Eventually all the schemas created through the process of interpreting a text must merge into a single graph (schema) which corresponds to the overall structure of the discourse for that text. The Figure 3-4 shows the single schema created for the discourse of the text in example 3.15.

The system does not save the schemas in any order, but recalls them as they are needed. That is, as a new sentence comes in, the inferencer tries to recall the right schema to attach it to. The order in which schemas are recalled follows: if the interpreted part of the discourse contains sentences S1,S2,...,Sn-1,Sn the schema which contains the last sentence will be recalled first. If the schema is not the right one, the system would recall the schema

[6] Throughout this thesis, by the term schema we mean instantiated schema.

which contains **Sn-1**, if **Sn-1** is not part of the previously investigated schemas. If no schema is found the sentence at hand will be added to the set of schemas as a schema with a single sentence. If the system succeeds in connecting the new sentence to the schema at hand , it attempts to connect the modified schema to the previous schemas in the discourse structure. A new sentence is connected to a schema if there exists an inferencing path between the sentence and a sentence in the schema. Consider again example 3.15 and Figures 3-3 and 3-4.

After failing to connect **b** to **a**, the system adds **b**, as a schema with a single sentence, to the discourse structure. Now the discourse structure contains two schemas containing **a** and the other containing **b**. To connect **c** to the discourse structure, the system first tries the second schema which turns out to be the right one. Then, the system tries to connect the second schema to the first through connecting **c** to **a** which fails. By connecting **d** to **c**, the system succeeds to connect **d** to the second schema. To connect the enlarged schema (containing **b**, **c** and **d**) to the first schema (containing **a**), the system connects **d** to **a**.

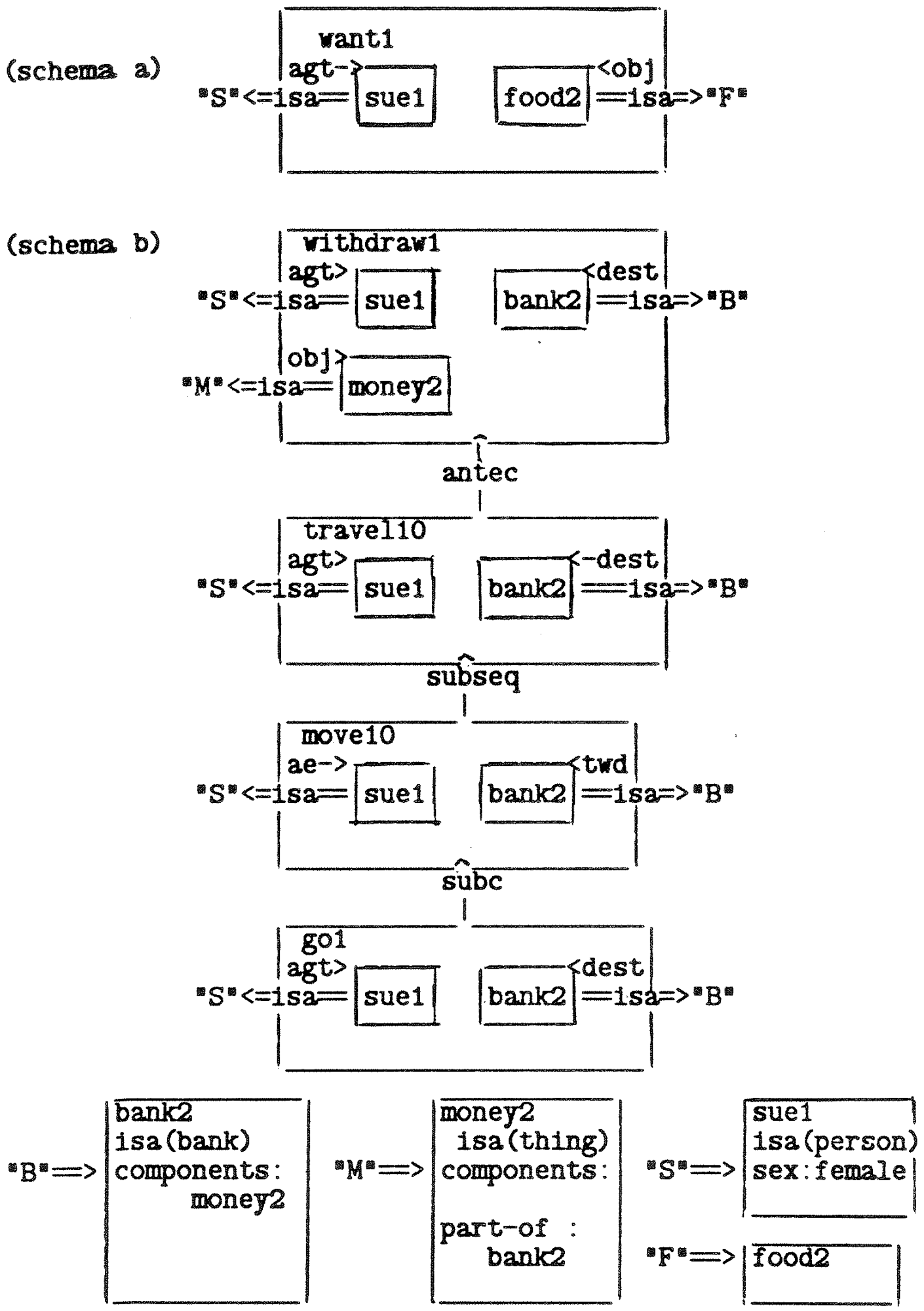

Figure 3-3: Withdrawing Money from the Bank

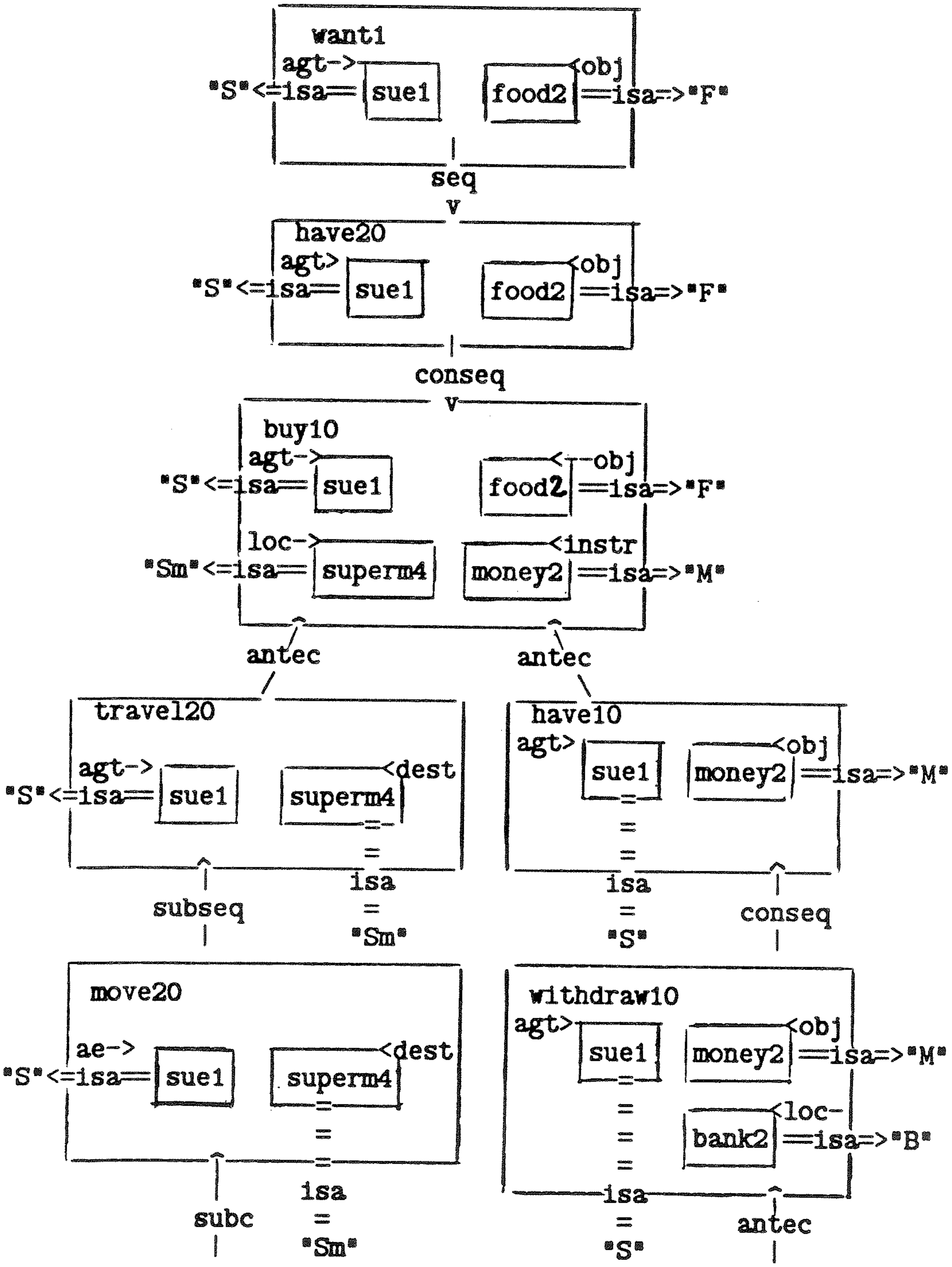

Figure 3-4: Getting Food

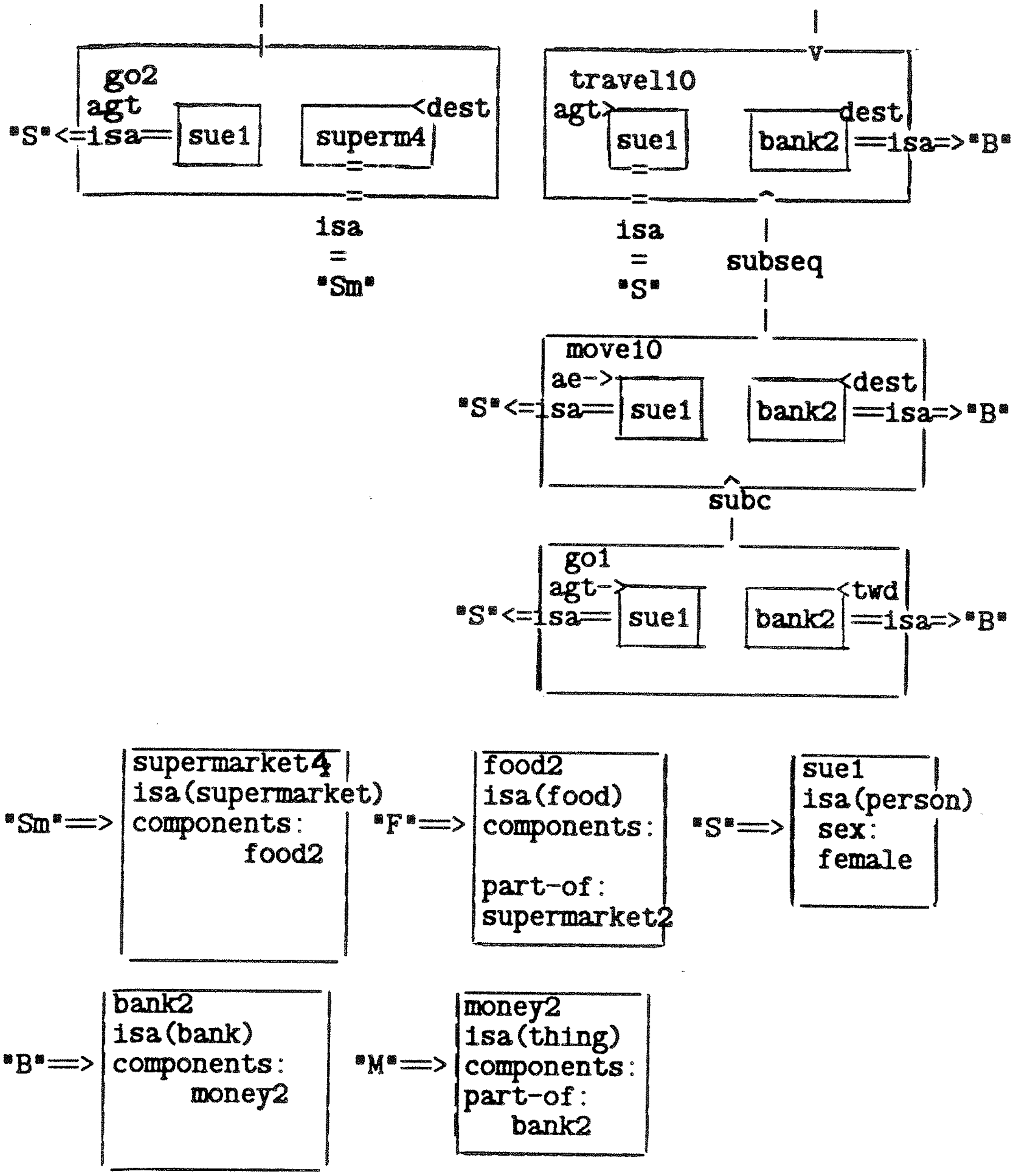

Figure 3-4: Continued

It seems so far CONNECTIVITY has made little use of the information associated with the conceptual structures of the items in two sentences for checking the coherence between them, but the reality is that the mechanism is based on the connectivity between conceptual structures. The system benefits from this information in two ways:

1. to support the likelihood that two sentences can be connected together,
2. to facilitate the process of connecting two sentences.

To show how frame information effects our connecting mechanism, we first need to define the notion of **expandability**.

3.2.3 Expandability

How can a writer or reader realize that two sentences are coherent? As Hobbs asks [Hobbs 78], why is it that if two sentences strike us as coherent, then they are coherent, otherwise they are not? Coherence implies relevance. That is, two coherent sentences are relevant. Two sentences are defined as relevant if they are expressing related topics; An incoming sentence is relevant to another sentence in the text if it is an expansion of that sentence. The new sentence is either describing an item in the earlier sentence or expanding on the idea centered around that item. Consider examples 3.16 and 3.17.

3.16.a Mary has some flowers
b They are red roses.

3.17.a Mary has some flowers
b she gathered them from the park.

In example 3.16, sentence **b** is relevant because it is describing the "flowers." Sentence **b** in example 3.17, describes how "Mary" got her "flowers;" "Gathering flowers" is relevant to "having flowers;" if somebody "gathers flowers" then she "has" them.

Focusing is the first step to writing relevant sentences [Sidner 83];

either in writing a sentence about an item or writing sentences which are the expansion of that sentence. In the case of a writer, focusing is the process through which he picks a topic to write about. To develop what he said in a sentence (to expand on it) he has to focus on an element of that sentence. That is, if the topic of the sentence is ▪flower▪ then the expansion of the sentence is something relevant to ▪flowers;▪ not something about the color of the hair of ▪Mary.▪ In turn, for the reader to recover the relevance between two sentences of the text, he must seek an item in an earlier sentence to match the idea described in the current sentence. In other words he tries to find what the focus of the earlier sentence was. In this way he only needs to look for certain kinds of information to connect the two sentences. Now, the point is this: how can we apply the human mechanism of focusing and relevance recognition to our inferencing mechanism?

TEXT-ANALYZER calls a topic the **expandable** item -- the item that a segment of a text is elaborating on[7] . An item in a sentence is considered the expandable item if the information associated with it can be used in process of connecting the sentence containing that item and another sentence in the text; information associated with the expandable item helps the inferencer to plausibly infer that another sentence in the text is an expansion of the sentence containing the expandable item. ▪Flower▪ in example 3.16 is considered the expandable item. To connect the two sentences, CONNECTIVITY, using the conceptual structure for ▪flower▪, reasons as follows:

1. ▪rose▪ is a type of ▪flower,▪

[7] The notion of expandable item does not imply that a sentence can have only one expandable item. An item in a sentence is considered expandable in regard to a segment of the text which is elaborating on that item; a sentence might have more than one item with such characteristics. In the following example, in sentence a, both ▪wedding▪ and ▪Hyde park▪ are considered expandable items. ▪Hyde park▪ is the expandable item in regard to the sentences b and c. ▪Wedding▪ is the expandable item in regard to the sentences d and e.

```
a John and Mary planned to have their wedding at Hyde Park.
b The park was one of the most beautiful parks
  in the whole area,
c which could hold a crowd of 500.
d The wedding was supposed to start at 5.00,
e and last until 7.00.
```

2. a "flower" has a type,

3. the second sentence is describing the type of "flower."

"Flower" is also the expandable item in example 3.17 because the event/state associated with flower can be used to establish an inferencing path between the two sentences. To connect the two sentences, CONNECTIVITY reasons that:

1. if "Mary" picks "flowers" then she has them. This inference is made by constructing the following path between picking -- the event associated with "flower" -- and "having."

```
pick [[agt,mary],[obj,flower]]
    seq
       have [[agt,mary],[obj,flower]]
```

2. as "Mary" picks the "flowers" she "gathers" them. This inference is made because:

```
gather [[agt,mary],[obj,flower],[loc,park]]
   subc
     pick [[agt,mary],[obj,flower],[loc,park]]
```

Formally, using the event/states associated with the expandable item, CONNECTIVITY performs the process of connecting two sentences in two steps: first a path is established between the sentence containing the expandable item to the event associated with it. second, a path is constructed between the second sentence and the already established path. In other words, the process of connecting two sentences is done forward from one sentence and backward from the other one. Consider example 3.18 and Figure 3-5.

```
3.18.a John went to a restaurant.
     b He ordered lobster.
```

the event associated with "restaurant" is eating which can be expressed as:

```
eat [[agt,customer],[obj,food],[loc,restaurant]]
```

The path between sentence **a** and eating is (step 2 in Figure 3-5):

```
eat10 [[agt,john1],[obj,food],[loc,restaurant3]],
 antec
   travel10 [[agt,john1],[dest,restaurant3]],
    subseq
      move10 [[ae,john1],[twd,restaurant3]],
        subc
          go1 [[agt,john1],[dest,restaurant3]]
```

Then the path between the sentence **b** and the above path is through its connection with eating event (step 3 in Figure 3-5):

```
eat10 [[agt,john1],[obj,lobster4],[loc,restaurant3]],
 subseq
   order1 [[ant,john1],[obj,lobster4]]
```

If sentences which CONNECTIVITY is trying to connect contain expandable items **I** and **J**, and **J** is a component (part) of **I**, then CONNECTIVITY would pick **I** to expand. In example 3.19, both "ice cream" and "kitchen" are expandable, but system picks "kitchen" as the expandable item, because "ice cream" may be a component of "kitchen" -- "ice cream" is kind of food and food is a component of "kitchen".

```
3.19.a Sally wanted some ice cream.
     b She went to the kitchen.
```

As side effect of the process of computing the expandable item (for connecting two sentences), we get two important results:

1. the expandable item determines which sentence is meant to be elaborated upon. In example 3.20, sentence **b** is elaborating on sentence **a**.

```
3.20.a. John went to a restaurant.
     b. He ordered lobster.
```

Now consider example 3.21:

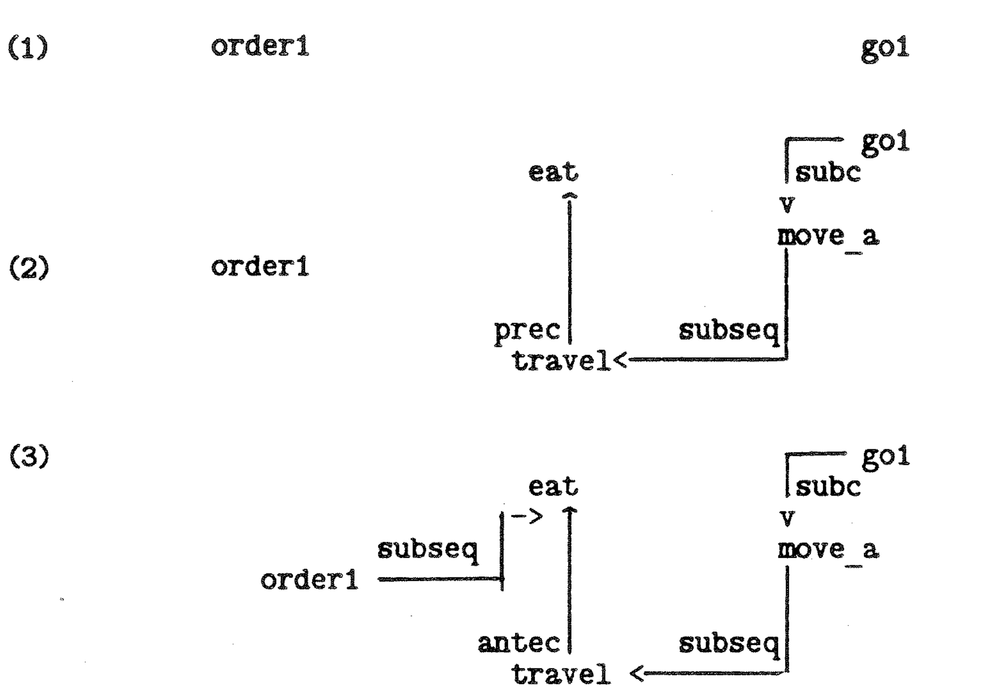

Figure 3-5: Process of Connecting two Sentences

3.21.a John was hungry.
b He went to a restaurant.

Here, the second sentence is elaborated upon which means the writer stated sentence a to introduce sentence **b**.

2. the expandable item can be used as a tool to find the interrelation between items in the sentence containing the expandable item, which in turn should effect the process of recovering the relationships between these items and those in other sentences. Consider the text shown in example 3.20, "John" is related to "restaurant" as a customer. This implies that for sentence **b** to be coherent with sentence **a**, "he" should refer to an entity which is in customer relation with "restaurant."

3.2.4 Minimal Connectivity Requirement

Let us review what we have said about the notions of coherence, relevance, expansion and expandable item. Two coherent sentences have cohesive ties, because they express related items. The cohesive ties are the relationships between those items in the two sentences which must hold in order that a certain coherence relation exists. The existence of cohesive ties between two sentences does not necessarily imply the existence of a coherence relation between them; however the lack of cohesive ties does imply the lack of a coherence relation. A sentence in coherence relation with another sentence is an expansion of that sentence. The expansion occurs in regard to an expandable item --- what is being focused on --- in another sentence. Such a sentence is either describing an item in the other sentence or elaborating on what is being said about that item in that sentence. We want to conclude that the relationship between the expandable item in a sentence and the items in another sentence can be used as an approximation to the likelihood of coherence between those sentences. This conclusion is the basis of the approach we took to prevent CONNECTIVITY from useless attempts to connect two sentences which are probably incoherent. The approach is based on the notion of minimal requirement for connecting two sentences which states that if there is no relation between any item in a sentence and an expandable item of the other sentence then there is no coherence between the two sentences. Consider examples 3.22 and 3.23.

3.22.a. Sally wanted some ice cream.
b. She went to the fridge.

3.23.a. Sally wanted some ice cream.
b. She went to the bank.

The first example satisfies the minimal connectivity requirement; there is a component relation between ■fridge■ which is the exapndable item and ■ice cream.■ The minimal connectivity requirement is not met in the case of example 3.23; there is no relation between ■ice cream■ and ■bank■ if either item is considered as the expandable item[8].

Here, we should mention that pronouns are not considered as expandable items. According to the definition of expansion a sentence is an expansion of another sentence in the discourse if it is adding some information about the topic of such a sentence; it is elaborating on the topic of that sentence. Such a sentence introduces the "things" that need to be known about the topic. The redundant "things" are presented by means of pronouns; pronouns carry no (new) information about a topic. Consider example 3.24.

```
3.24.a John entered the hotel.
     b It was a 24 story building.
```

The second sentence is given to assert that the "hotel" is a "24 story building." The pronoun "it" is referring to "hotel" which the writer avoided repeating in the second sentence.

Now consider example 3.25. The second sentence does not introduce any new item related to the "glass" which is the topic of the first sentence. Sentence **b** elaborates on the event/state described in sentence **a**; it gives more detail about what happened to the "glass."

```
3.25.a John dropped the glass.
     b It broke.
```

When CONNECTIVITY encounters a sentence with no non-pronoun item (like sentence b in above example), it constructs an inferencing path between it and the sentence containing the expandable item.

[8] It seems that applying the notion of expandable item and minimal connectivity requirement is not suitable in the sentences with many arguements because in this case the inferencer might have to try too many items. Considering some kind of ordering, for example, first agent and affective entity, then location simplifies this problem.

3.2.5 Minimal Requirement for Connecting a Sentence to a Schema

An instantiated schema corresponds to a segment of the text which describes a single topic. That is, a schema covers a set of sentences which are connected together according to the connection between their topics. The major topic which brings together the topics of different sentences is called the focus of the schema covering the sentences. We define the focus of a schema as:

> **the last expandable item** for which the expansion of its associated event/states results in a coherence path between an incoming sentence and the schema.

Consider the example 3.26. Because the expandable item (focus) is computed when there exists a connection between two sentences, a schema must cover more than one sentence in order to have a focus. Therefore, the focus of the schema containing sentence **a** is empty. The focus of the schema containing sentences **c** and **b** is "bank." The focus of schema created from connecting sentence **d** to the schema containing sentences **b** and **c** is "supermarket." The focus of the single schema which connects the last schema to sentence **a** is again "supermarket" which means the whole text is about "Sue's going to the supermarket."

```
3.26.a Sue wanted some ice cream.
     b She went to the bank.
     c She withdrew some money.
     d She went to the supermarket.
```

To check the minimal connectivity requirement between a sentence and a schema, CONNECTIVITY works as follows:

1. it checks to see if there is any relation between the focus of the schema and any item in the sentence at hand. If that is the case, then the system tries to establish a coherence path between the sentence and the schema. In example 3.24, there is a relation between "ice cream" and "supermarket" which is the focus of the second schema. Therefore, CONNECTIVITY will try to connect the sentence to the schema.

2. If there is no relation between the focus of the schema and the items of the sentence at hand, then CONNECTIVITY checks the relation between the items of the current sentence and the items appearing in the schema in the following manner: if the text contains sentences S1,...,Sj,...Sk,...Sm,...Sn-1, and the schema contains sentences Sj, Sm, Sk, then the items of the current sentence will be checked against the items that occurred in sentence Sm. If Sm is not the right one the CONNECTIVITY considers Sk. Consider again example 3.26. The focus of the second schema is "bank" which has no relation with "supermarket." Therefore CONNECTIVITY will compare item "money" in sentence **c** with "supermarket." Because money is a component of "supermarket," "supermarket" is expanded to establish a schema to which sentence **c** must find a path.

3.3 Detailed Examples

In this section we discuss the result of experimenting with TEXT-ANALYZER on three sample texts.

3.3.1 Restaurant Story

```
John was hungry.
he went to a restaurant.
the waiter seated him.
he gave him the menu.
he came to John's table.
he ordered lobster.
he was served.
he tipped the waiter.
he left.
```

Figure 3-7 shows the input (the parsed text) to the system and Figure 3-8 shows the discourse representation that it produced as output.

```
text([
     [be,[agt,[john,[mod,hungry]]]],
     [go,[agt,he],[dest,restaurant]],
     [seat,[agt,waiter],[ae,he]],
     [give,[agt,he],[recp,john],[obj,menu]],
     [come,[agt,waiter],[dest,[table,[of,john]]]],
     [order,[agt,john],[obj,lobster]],
     [serve,[benf,he]],
     [tip,[agt,he],[recp,waiter]],
     [leave,[agt,he]]]).
```

Figure 3-6: The Case Format for the Restaurant Story

Here, we will explain some of the inferences that the system makes in interpretation of this text.

1. If somebody is ▪hungry▪ ▪he▪ usually ▪goes▪ to a place (for example restaurant) that he can get something to eat. Therefore ▪John▪ who was ▪hungry▪ must be the one who ▪went▪ to the ▪restaurant.▪
2. In order for ▪John▪ to eat something he first had to be ▪seated▪ by the ▪waiter.▪
3. In order for ▪john▪ eat something he was ▪seated▪ by the ▪waiter▪ and ▪given▪ the ▪menu▪ (so he can order what he wants to eat.)
4. The ▪waiter▪ must ▪come▪ to the ▪John's table▪ so ▪John▪ can ▪order▪ what (▪lobster▪) he wants to eat.
5. The ▪waiter serves John▪ the ▪lobster▪ (he had ordered) so ▪he▪ (John) can eat it.
6. ▪John tipped the waiter▪ because he liked the ▪lobster▪ he (John) had eaten.
7. After ▪John▪ ate ▪lobster▪ he departed from the ▪restaurant▪ to ▪leave▪ that place.

Notice that:

1. the relationship between ▪John▪ and ▪restaurant▪ is reflected through the ▪customer▪ relation.
2. ▪lobster▪ and ▪menu▪ are related to ▪restaurant▪ as ▪props.▪
3. the relationship between ▪waiter▪ and the restaurant is ▪employee.▪

```
lobster1
  isa(food)
  with(
        [prop,[of,restaurant1]])
  actions(
        [eat,[obj,lobster1]])


restaurant1
  isa(restaurant)
  with(
        [prop,lobster1,menu1,money]
        [employee,waiter1,cashier]
        [furniture,table1,light]
        [customer,john1])
  actions(
        [eat,[agt,john1],[obj,lobster1],[loc,restaurant1]])


waiter1
  isa(waitperson)
  with(
        [employee,[of,restaurant1]]
        [sex,male])
  actions(
        [serve,[agt,waiter1],[obj,food],[recp,customer]])


table1
  isa(table)
  with(
        [furniture,[of,restaurant1]]
        [of,john1])
  actions()


menu1
  isa(menu)
  with(
        [prop,[of,restaurant1]])
  actions()
```

Figure 3-7: Discourse Structure for the Restaurant Story

```
john1
  isa(person)
  with(
       [customer,[of,restaurant1]]
       [sex,male])
  actions()



    eat10 [[loc,restaurant1],[agt,john1],[obj,lobster1]]
    seq
        depart10 [[source,restaurant1],[agt,john1]]
        subc
            leave1 [[agt,john1]]
    coord
        like10 [[obj,lobster1],[agt,john1]]
        seq
            tip1 [[recp,waiter1],[agt,john1]]
    subseq
        serve1 [[agt,waiter1],[benf,john1],[obj,lobster1]]
    subseq
        order1 [[obj,lobster1],[agt,john1]]
        prec
            travel20 [[dest,table1],[agt,waiter1]]
            subc
                come1 [[dest,[table1,[of,john1]]],[agt,waiter1]]
    subseq
        seat1 [[ae,john1],[agt,waiter1]]
        subseq
            give1 [[obj,menu1],[recp,john1],[agt,waiter1]]
    prec
        hungry1 [[agt,john1]]
    antec
        travel10 [[dest,restaurant1],[agt,john1]]
        subseq
            move_a10 [[twd,restaurant1],[ae,john1]]
            subc
                go1 [[dest,restaurant1],[agt,john1]]
```

Figure 3-7: Continued

3.3.2 Coconut Palm

```
The coconut palm appeared on a tropical island.
Its seed which is the coconut of commerce is protected
by its fibrous husk from waves of ocean.
It is adapted to it.
It is distributed by  waves of ocean.
It is carried over a thousand miles,
It grows on a distant shore.
```

Figure 3-9 shows the input text and Figure 3-10 illustrates the discourse structure produced for the text.

```
text([
      [appear, [ae, [palm, [type,coconut]]],
                                 [loc, [island, [type,tropical]]]],
      [protect, [ae, [seed, [of,it],
                            [type, [coconut, [type,commerce]]]]],
                  [instr, [husk, [type,fibrous], [of,it]]],
                    [from, [wave, [of,ocean]]]],
      [adapt, [ae,it]],
      [distribute, [ae,it], [instr, [wave, [of,ocean]]]],
      [carry, [ae,it], [range, [mile, [num,thousand]]]],
      [grow, [ae,it], [loc, [shore, [type,distant]]]]
                                                    ]).
```

Figure 3-8: The Case Format for the Coconut Palm Text

The inferences that the system made to connect sentences of text together are:

1. As "palm1" grows (on a "tropical island") it "appears" there.
2. A plant as it grows starts producing "seeds" (as it is the case of "palm1" which produces "seed1").
3. After "seeds" are produced, they are "distributed" by the "waves of ocean."
4. While the "seed" is being "distributed" by "waves of ocean," it "protects" itself from it.
5. As the "seed" "protects" itself against "waves of ocean" it is "adapted" to it.
6. For the "seed" to be "distributed" on a "distant shore" "it" must be "carried" there.
7. For the "seed" to "grow" on a "distant shore" it must be sowed there first.
8. The "seed" is sowed in a "distant shore," after it is "carried" there.

The frame structures show that:

1. type of "palm1" is "coconut" and it is located in a "tropical island" (as it is a component of that island), and it has "seed1" as its seed.
2. "island1" which is "tropical" has "palm2" as one of its components,
3. "seed1" whose type is "coconut of commerce" is a seed of "palm1" and "husk1" is one of its parts,
4. "husk1" which is part of "seed1" is "fibrous."
5. ocean1 has wave1 (wave1 is a part of ocean1).

```
seed1
  isa(seed)
  with(
        [seed,[of,palm1]]
        [parts,husk1]
        [type,[coconut,[type,commerce]]]
        [shape,shape])
  actions(
        [distribute,[obj,seed1]])

palm1
  isa(plant)
  with(
        [seed,seed1]
        [component,[of,island1]]
        [loc,island1]
        [type,coconut]
        [parts,branch,root]
        [height,height]
        [fruit,fruit])
  actions(
        [grow,[ae,palm1],[loc,island1]])

shore1
  isa(thing)
  with(
        [parts,[of,ocean1]]
        [type,distant])
  actions()

island1
  isa(island)
  with(
        [component,palm1]
        [type,tropical])
  actions()

wave1
  isa(thing)
  with(
        [parts,[of,ocean1]])
  actions()
```

Figure 3-9: Discourse Structure for the Coconut Palm Text

```
ocean1
  isa(ocean)
  with(
       [parts,shore1,wave1,water])
  actions()

husk1
  isa(husk)
  with(
       [parts,[of,seed1]]
       [type,fibrous])
  actions()


    distribute1 [[range,[mile,[num,thousand]]],[ae,seed1],
                                 [instr,[wave1,[of,ocean1]]]]
    coord
        carry1 [[dest,shore1],[ae,seed1],[instr,wave1],
                                 [range,[mile,[num,thousand]]]]
        prec*
            sow20 [[ae,seed1],[loc,shore1]]
            antec*
                grow1 [[loc,[shore1,[type,distant]]],[ae,seed1]]
    coord
        protect1 [[ae,seed1],[from,[wave1,[of,ocean1]]],
                  [instr,[husk1,[of,seed1],[type,fibrous]]]]
        subseq*
            adapt1 [[to,wave1],[ae,seed1]]
    seq*
        produce10 [[obj,seed1],[ae,palm1]]
        coord*
            grow10 [[ae,palm1],[loc,island1]]
            coord
                appear1 [[loc,island1],[ae,palm1]]
```

Figure 3-9: Continued

3.3.3 Rocket Story

```
The V2 rocket stood in the New Mexico desert.
It weighed 5 tons.
For fuel, it carried eight tons of oxygen.
Everything was ready.
Scientists withdrew behind earthmounds.
A red flare rose.
The rocket was fired.
It rose.
Radar tracked it.
It plunged to the earth.
```

Figure 3-12 shows the set of instantiated frames and event/states sequence that is produced as a discourse structure for the text.

```
text([
      [stand,[ae,[rocket,[type,v2]]],
                          [loc,[desert,[type,newmexico]]]],
      [weigh,[obj,it],[msr,[ton,[qty,five]]]],
      [carry,[ae,it],[pu,[fuel,
           [com,[oxygen,[msr,[ton,[qty,five]]]]]]]],
      [ready,[ae,everything]],
      [withdraw,[agt,scientist],[dest,earthmound]],
      [rise1,[ae,[flare,[color,red]]]],
      [fire,[ae,rocket]],
      [rise2,[ae,it]],
      [track,[instr,radar],[obj,it]],
      [plunge,[ae,it],[dest,earth]]
                                     ]).
```

Figure 3-10: The Case Format for the Rocket Story

The important inferences that the system makes in order to interpret the text are:

1. ▪rocket1▪ of type ▪v2▪ which ▪weighed 5 tons▪ was ▪standing▪ in a ▪desert▪ in ▪New Mexico▪ ▪carrying 8 tons▪ of ▪fuel1▪ (refer to the frame structures for rocket1 and fuel1),
2. for ▪rocket1▪ to make a flight from the ▪New Mexico desert▪ it takes off from there,
3. in order for ▪rocket1▪ to take off it must be ▪standing▪ somewhere (for example the ▪New Mexico desert▪),
4. before the ▪rocket1▪ can take off it (rocket1) must be ▪ready,▪
5. the ▪red flare▪ (flare1) that ▪rises▪ from the ▪rocket▪ is a ▪signal▪ for the rocket's taking off,
6. ▪rocket1▪ is ▪fired▪ to take off from the ▪New Mexico desert,▪
7. the rocket (rocket1) ▪rises▪ as it ascends from the ▪New Mexico desert,▪
8. the ▪scientists▪ ▪withdraw▪ behind ▪earthmound1▪ to protect themselves from ▪rising▪ of ▪rocket1,▪
9. one of steps in flight of the ▪rocket▪ from a source (for example the New Mexico desert) to the destination (for example earth1) is to ▪descend▪ to ▪earth1,▪
10. a rocket ▪plunges▪ into the earth (the destination of the flight) as it decsends.

```
earth1
  isa(thing)
  with()
  actions()

radar1
  isa(radar)
  with(
        [prop,rocket]])
  actions(
        [track,[obj,rocket],[instr,radar1]])

rocket1
  isa(rocket)
  with(
        [prop,[flare1,[color,red]]]
        [observer,scientist1]
        [fuel,[fuel1,[com,[oxygen,[msr,[pound,[num,five]]]]]]]
        [weight,[ton,[num,eight]]]
        [loc,desert1]
        [type,v2])
  actions(
        [flight,[ae,rocket1],[source,desert1],[dest,earth1]])

flare1
  isa(thing)
  with(
        [prop,[of,rocket1]]
        [color,red])
  actions()

scientist1
  isa(person)
  with(
        [observer,[of,rocket1]]
        [sex,male])
  actions()

earthmound1
  isa(thing)
  with()
  actions()
```

Figure 3-11: Discourse Structure for the Rocket Story

```
fuel1
  isa(thing)
  with(
       [com,[oxygen,[msr,[pound,[num,five]]]]])
  actions()

desert1
  isa(thing)
  with(
       [type,newmexico])
  actions()


    takeoff10 [[ae,rocket1],[source,desert1]]
    subseq
        fire1 [[ae,rocket1]]
    antec
        signal10 [[ae,rocket1],[instr,flare1]]
        subc
            rise1 [[ae,[flare1,[color,red]]]]
    antec
        ready1 [[ae,rocket1]]
    prec
        stand1 [[loc,[desert1,[type,newmexico]]],
                               [ae,[rocket1,[type,v2]]]]
    subseq*
        flight10 [[ae,rocket1],[source,desert1],
                                   [dest,earth1]]
        subseq
            descend10 [[dest,earth1],[ae,rocket1]]
            subc
                plunge1 [[dest,earth1],[ae,rocket1]]
        coord
            track1 [[obj,rocket1],[instr,radar1]]
        subseq
            ascend10 [[source,desert1],[ae,rocket1]]
            subc
                rise2 [[ae,rocket1]]
                coord
                    protect10 [[ae,scientist1]]
                    subseq
                        withdraw1 [[dest,earthmound1],
                                            [agt,scientist1]]
```

Figure 3-11: Continued

Chapter 4

Related Work

Recent study in natural language understanding has concentrated on the relation of discourse structure to contextual references, and how analysis of discourse can aid in resolving references. In this chapter, we will discuss some approaches which follow this line of research and their relation to our work.

The approaches we are going to discuss can be classified as those which use prior knowledge about the discourse structure, and those in which the discourse structure is built up dynamically.

Both approaches have one thing in common: they agree that a sentence should not be interpreted in isolation from the rest of the established discourse for a text. The major difference between them is in their dependencies on the representation of the knowledge base and of the text. The first approach tends to ignores the structural effects of the text itself on the interpretation of a sentence in favor of using some prior knowledge about the conceptual structure about what is described in the text. In contrast to this approach is the one in which interpreting a text is the process of dynamically constructing the discourse structure. In this approach, it is assumed that the structural clues (for example the interrelations between the sentences) in the text itself are sufficient for such construction.

Another difference between the two approaches is the way in which each employs the inferencing mechanism. In the first approach, the set of inferences required for resolving a reference is implied through the prestored dependencies. Therefore, the process of reference resolution is reduced to a matching process. In the second approach, the inferencer has to create dynamically the set of inferences required to resolve references.

4.1 Pre-defined Discourse Structure

This approach uses scripts, frames and schemas which correspond to discourse structures that the text would potentially have. These schemas consist of sets of conceptually unified propositions about some major concept(s) that the text is about. In this approach, the interpretation of the text corresponds to the process of matching against or filling in the slots of schemas with the units of the text.

This approach can be divided into two classes. First, the one which ignores the discourse structure of the text. It assumes that the logical relationships among the sentences of the text is reflected implicitly in the script or frame structure. Examples of this class are frames [Minsky 75], and scripts [Schank 77].

The second class uses the discourse structure of the text represented as the explicit relations among its sentences. Examples are task-oriented dialogue, [Grosz 77], and PAL [Sidner 78].

4.1.1 Scripts

Schank and his associates at Yale introduced the idea of scripts interpreted as stylized stories about the certain concepts that a text is about. Scripts are data structures composed of a set of templates. These templates define the set of events[1] described in the scripts and some generalized arguments of each event. For example a script for "visiting a doctor's office" has a structure like the one shown in figure 4.1. Attached to each template is a set of constraints. A system based on scripts uses a pattern matcher which matches the input sentence against the templates defined in the scripts. The set of constraints attached to each template are checked while the matching process takes place.

A script defines a structureless discourse. In other words, the set of entities and event/states associated with a script have implicit (conceptual)

[1] the set of events which are usually associated with the concept that the script represents

```
Script: Doctor's office
Track:  hospital
Props:  waiting room, visiting room, prescription,...
Roles:  visitor, doctor, secretary,...
Entry condition: visitor has money, visitor is sick,...
Results: visitor has less money, doctor has more money,...
Scenes: entering, waiting, being examined, paying the fee,...
```

Figure 4-1: Visiting Doctor's Office

rather than explicit connection. Because of the structureless nature, for a script to be applicable, a close match must be made between its units and units of a text. Scripts can be used in processing a certain class of texts, namely those texts which follow the line dictated by the script. Any shift of the text from that line might cause errors in interpretation of the text.

Schank considered scripts as a tool to model human memory organization. Later, he generalized the idea of scripts into the idea of dynamic memory, [Schank 79]. According to this idea, the script of a given situation must be built up dynamically (or in Schank's words reconstructed) rather than being retrieved from memory. Information in memory is organized into chunks called memory organization packets (**MOPS**). A MOP is really a script which can be applied to a variety of situations. For example, instead of having a different script for ■visiting a lawyer's office■ and ■visiting a doctor's office,■ there is only a MOP for ■visiting professional office.■ The MOP for ■visiting a professional office■ is instantiated when the script for ■visiting■ a particular ■professional office■ is required. Now the process of understanding a given situation is finding the relevant MOPS. The script for understanding a text corresponds to a superscript constructed from MOPS rather than one recalled entirely from memory. For example the set of MOPS that provides the necessary information for creation of a script for ■visiting a doctor's office■ would be something like a MOP for ■professional office visit,■ ■doctor,■ ■contract■, ■health care■, etc.

Smith, in the approach described below, uses a similar notion

-- constructing a structure for the discourse out of a smaller set of conceptually dependent structures (conceptual instances) stored in the knowledge base.

4.1.2 Smith's work

The goal of this approach is to illustrate how employment of a taxonomically organized knowledge base of related concepts in the interpretation of a text can result in resolution of a broad range of references present in the semantic interpretation of a text. To Smith, reference is a mapping process which takes a word from a sentence and finds its relation to the concepts present in the context until it reaches the concept which acts as the base concept for the item. For example, in the following text[2], "employee" will be related to the concepts "NBC" as its employee, to "settlement" as the recipient, to "discrimination" as the one who is discriminated against, and eventually to the concept "person" which is the base concept for employee.

```
4.2.a NBC agreed to pay $2 million
    b to settle a sex discrimination suit
      brought by women employees.
```

The context is an integrated set of conceptual structures which were activated during the interpretation of the previous part of the text. The process of constructing such a context can be viewed as the expansion of the concepts which either implicitly or explicitly are associated with the lexical items in the text. If we call this context the active (local) context, then the global context is the set of all those prestored dependencies (in the knowledge base) between the concepts present in a text which might be required for integration of that text. Then, the active context filters these dependency relations to exploit those which are required at any instance of the interpretation of the text. The drawback of this approach is its exclusive reliance on the organization of the knowledge base. This makes the approach inflexible for handling of texts whose organizations do not fit the prestored dependencies (in the knowledge base).

[2] one of the texts examined by Smith.

4.1.3 Task-oriented Dialogue

Grosz uses this approach to illustrate that the structure of the dialogue that a speaker carries on corresponds to the structure of the major topic that the dialogue is about. She particularly emphasizes that the speaker's focus of attention at any instance of the dialogue is reflected in the structure of the topic (of the dialogue). The structure of the topic is to represent the focus space of the dialogue at any instance of processing.

In her approach, the topic of the dialogue is a single major task: the assembly of a compressor. The task, in turn is divided into subtasks (and subtasks into subsubtasks). The structure of the task is similar to the tangled hierarchy shown in Figure 3.1 (chapter 3). The root of the tree is the major task which is also considered as the initial focus of the dialogue; its branches are subtasks. If the current sentence is about a subtask denoted by some node I, then the immediate focus for the next part of the dialogue would be I. Subtasks of I are considered as the implicit focus, elements to which the conversation is likely to shift. The set containing a task and its subtasks is called the open (active) focus as long as the conversation is centered around them. Otherwise a shift in focus has occurred and the set will become a part of the closed focus (which might reopen in later conversation). Grosz uses this structure to search for the antecedents. The search starts from the open focus. If the antecedent is not there (a shift in focus), the search continues in the closed focus[3].

There are some problems with the Grosz approach. First, not every text can easily be defined in a structured format although task-oriented dialogues usually follow a certain structure. Second, limiting the discourse of the text to such a structure limits the speaker to focus only in the directions dictated by the structure.

In spite of these drawbacks, the Grosz' approach is important because of its notion of focus space. Focus has direct effect on references, and it would be a useful tool for resolving references but using a predefined structure for the focus is not an adequate solution. Unfortunately, dynamic computation of focus

[3] also a shift in focus has occurred if the focus changes from a task/subtask to one of its subtasks/subsubtasks.

is not always possible either, because it requires prior knowledge about the sentence whose references are to be resolved (see chapter 3.).

4.2 Dynamic Computation of the Discourse -- Discourse Cohesion Based Approach

This approach takes advantage of the interrelationships between the sentences of a text. The idea is that, the knowledge about the coherence relation between an incoming sentence and a sentence in the preceding discourse is sufficient for the interpretation of the sentence[4]. The coherence relations are generalizations over the ways that the discourse of a text flows. Examples of the approaches which follow this line are by Hobbs [Hobbs 78], Lockman and Klappholz [Lockman 83], and Alterman [Alterman 82]. All these approaches have a similar framework. That is, to interpret a sentence, the inferencer, generates a set of inferences which plausibly confirm the existence of a coherence relation between the current sentence and a sentence in the established discourse. The references are resolved as part of this process.

4.2.1 Alterman, Conceptual Coherence Discourse

Alterman's NEXUS presents a theory of event/state conceptual coherence. NEXUS is supplied with a dictionary compiled with a set of connected event/states. The connection between each pair of concepts can be any of the seven conceptual coherence relations that Alterman proposes. NEXUS uses the dictionary to connect an incoming sentence to a sentence in the preceding discourse by establishing an inferencing path between the two sentences. The path is composed of a set of conceptually coherent event/states (according to his dictionary) which are present in the text either explicitly or implicitly.

Our system functions much the way that NEXUS does, mainly because we use NEXUS' set of coherence relations and the subset of MAINSTR, MAINR

[4] see chapter 3 for more detail

which is used for coherency recognition between two sentences is similar to NEXUS' dictionary. But TEXT-ANALYZER differs from NEXUS in some important aspects.

NEXUS and TEXT-ANALYZER have different notions about the ▪connection▪ between two sentences. To NEXUS, connection between two sentences is equivalent to copying an appropriate portion of the dictionary. To TEXT-ANALYZER, a connection between two sentences exists if one of them is an ▪expansion▪ of the other. In order to establish a **▪connection▪** (in a way that is interpreted by the system) between two sentences, TEXT-ANALYZER has to deal directly with the knowledge associated with the items of the sentences. To NEXUS, the nature of these items is of little or no importance. This difference between the two systems is most apparent in the representation that each creates for the discourse of the text. NEXUS' representation of discourse is summarized by event/state trees, while in TEXT-ANALYZER the detailed relationships between the items, reflected through the frame structures for them, are part of the discourse representation. In other words, in this representation the result of reference resolution can be traced in the frame structure of the items of the two sentences. These structures (frames), besides being helpful in interpreting later parts of the text, make the discourse structure more informative for later applications, for example, questioning the discourse structure. Consider the following example:

4.3.a Mary needed some flowers.
b She went to the flower-shop across the street.
c She bought some red roses.

Assume one asks the following question:

Where is the flower-shop?

The answering system using the representation that TEXT-ANALYZER produces for the discourse need not to look for the proper sentence to find the location of ▪flowershop.▪ In process of constructing the discourse, the location of the ▪flowershop▪ has been found and reflected in its conceptual structure and it can be easily recovered. Presence of the conceptual structures (of the items) in the discourse representation is helpful in answering questions of following type:

```
What is the color of Mary's flowers?
What is the type of Mary's flowers?
```

The answering system dealing with the discourse structure that NEXUS produces (for a text) has to use a separate set of axioms to find the correct solutions. Axioms are in the form:

```
    isa(rose,flower)
    color(rose,red)
    isa(X,Y),isa(Y,Z) ----> isa(X,Z)
```

This is a redundant process because NEXUS had previously used the same set of axioms while trying to connect the sentences of the text together. TEXT-ANALYZER by saving whatever it learns about an item under its conceptual structure while constructing the discourse structure frees the answering system from such a redundant process.

By elaborating on the notion of "expansion," TEXT-ANALYZER can derive the notion of "item expansion" and therefore justify the definition of a **detail** relation. NEXUS is unable to recover such a relation between sentences of a text; NEXUS can only recover the coherence relation between those sentences which are "logically" connected according to the event/state described by them. For example, NEXUS would not be able to realize the second sentence of the following text is expanding on the item "flower" of the first one.

```
   4.4.a Mary bought some flowers.
       b They were red roses.
```

TEXT-ANALYZER takes advantage of the notion of "expansion" in two other ways. First, through the notion of "expandable item" TEXT-ANALYZER is able to derive the interrelations between the items of a sentence which are important in the process of interpreting the sentence. Recovering this type of relation is ignored by NEXUS. For example, in the case of the example 4.5, NEXUS would not account for the relationship between "Sally" and "supermarket" as customer.

```
4.5.a Sally went to the supermarket.
    b She spent $100.
```

The second way that TEXT-ANALYZER benefits from the notion of expansion is by extracting the idea of minimal requirement for connecting two sentences. By employing this idea, TEXT-ANALYZER's inferencer has to trace a connection between some items in a sentence and an expandable item in another sentence. In this way the inferencer is prevented from the wasteful process of connecting two sentences whose connection is not apparent (or not possible). Alterman recognizes the necessity of having a mechanism to check the existence of some kind of relation between the items in the current sentence and the items in the preceding discourse, but abandons the idea due the complexity that it would add to his system.

4.2.2 Hobbs' approach

Hobbs' approach is based on recovering a rhetorical relation between sentences of a text. Rhetorical relations are coherence relations which mainly deal with the style that a writer follows in writing of a text in contrast to conceptual coherence relations which deal with the way that people conceptually arrange event/states or relate them together.

In Hobbs' system each rhetorical relation is defined in terms of a set of rules which must be satisfied in order to conclude that such a relation holds between two sentences. For example, **elaboration** is defined as:

> ... sentence **S1** is an **elaboration** of sentence **S0** if some propositions **P** follow from the assertions of both **S0** and **Si**, but **S1** contains a property of one of the elements of **P** that is not in **S0**.

The system is also supplied with a knowledge base which is compiled of facts about the *things* in the real world. For example, it has knowledge about *safe* and *combination,* and it knows that to *open* a *safe* one should *dial* its *combination.*

To recover a coherence relation, the system tries by using the axioms and rules of inference in the knowledge base to satisfy the set of rules required in the definition of a coherence relation. For example, the system would realize the coherence relation between two sentences in text 4.6 as **elaboration** through the following steps:

1. because John can open the *safe* therefore he must *know* what to *do* in order to *open* the *safe.*
2. the knowledge associated with *combination* says that somebody, for example, *he* must *know* the *combination* in order to be able to *open* the *safe.*

```
4.6.a John can open the safe.
    b He knows the combination.
```

By comparing the two derived results, the system infers that *John* must be *he*, and *combination* must be related to *safe.*

TEXT-ANALYZER functions in a similar way. It has a knowledge base which asserts what concepts (particularly what actions) are usually associated with an entity. The system tries to take advantage of the knowledge associated with the entities in the sentences to facilitate the connection between two sentences.

Let us see how our system would handle the text of example 4.6. First it would instantiate the frame structures for the items introduced in the two sentences. These frames are shown in figure 4.2 (Note that the figure shows the status of the frames after processing the text). TEXT-ANALYZER realizes *safe* as the expandable item[5] in contrast to Hobbs' system where the expandable item is *combination.* The event associated with *safe* states

[5] because a combination is part of a safe.

that the "instrument" of opening a "safe" is its "combination." Then the system tries to find a path between the following sentences:

```
open [[agt,john1],[ae,safe2],[instr,combination],[modal,can]]

know [[agt,he],[ae,combination3]]
```

To connect these two the following path is established between them.

```
open [[agt,john1],[ae,safe2],[instr,combination3]]
   prec
      dial [[agt,john1],[ae,combination3]]
         antec
            know [[agt,john1],[ae,combination3]]
```

which means:

1. one of the ways to open a "safe" is "dialing" its "combination,"
2. to "dial" a combination one must "know" the "combination."

4.2.3 The Lockman & Klappholz approach

The inferencing mechanism of Lockman and Klappholz [Lockman 83] is based on the notion of "expansion." That is, each new sentence in a text is considered to be an expansion of another sentence in the preceding discourse. The inferencer's task is to find a "fit" for the incoming sentence in the "context graph" which represents the so far interpreted discourse. By "fit" they mean that for some current sentence, **Sj**, there is another sentence **Si** in discourse structure such that Expand(Si,Sj) is true and i<j. "**Expand**" is a predicate that tests for different types of possible "expansion" between two sentences; expansions are defined in terms of a set of rules which satisfy the definition of coherence relations between two sentences. To find a "fit" for a new sentence

```
safe2
      isa(safe)
      with([[parts,combination3],
            [size,size]],
      actions([[open,[ae,safe2],[instr,combination3]])


combination3
      isa(thing)
      with([[parts,[of,safe2]]])
      actions([dial,[ae,combination3]])
```

Figure 4-2: Frame Structures for Safe and Combination

the inferencer generates as many assumptions (inferences) as possible to derive the definition of one of the coherence relations from the knowledge associated with the items in two sentences. It will continue the process of generating assumptions until either it can prove the existence of a coherence relation between two sentences, or it has exhausted its inferences. In the latter case, the inferencing mechanism would consider another coherence relation. If the inferencer fails to prove the existence of any coherence relation between two sentences, it will move up the context graph to find the next suitable sentence(node) and attempt to connect to the new sentence.

TEXT-ANALYZER has a similar notion of coherence relation and its interpretation in terms of expansion. Our system also considers the notion of "fit" to connect a new sentence to the preceding discourse and the TEXT-ANALYZER representation of the discourse structure is a graph similar to the one that Lockman an Klappholz suggest in their system.

However, we do not agree with Lockman and Klappholz in some points. First of all, it is undesirable to have the inferencer generate as many inferences as possible in order to find a connection between two sentences. This is particularly wasteful if there is no apparent connection between two sentences.

Second, we consider the relationship between the items of the current sentence itself. The recovery of this relationship may have a direct effect on the interpretation of this sentence. It seems that feature is missing in the system of Lockman and Klappholz.

In this chapter we have discussed two different classes of approaches to discourse analysis. In our approach, we tried to integrate the better features of each class and avoid their drawbacks. Although, we do not agree with the approaches in which text interpretation is solely the processing of predefined structures, we use the idea of predefined structure (in the form of frames) to assist the process of discourse construction. TEXT-ANALYZER mainly follows the line of the second class of approaches. TEXT_ANALYZR uses the notions of *expandability* and *minimal connectivity requirement* to reduce the inefficiency that the inferencing mechanism in this class of approaches usually has which is the sometimes unnecessary search for recovering the coherence relation between two sentences.

Chapter 5

Conclusion

5.1 Summary

In the previous chapter, we discussed several discourse analysis approaches each of which offered a different insight into aspects of the use of discourse structure in the process of interpretation of a text, in general, and contextual reference resolution, in particular. Our research is an attempt (an experimental model) to integrate some of these insights into a single model.

Our approach is based on the theory of discourse coherence: A coherent discourse has the property of exhibiting structural relationships between its different sentences. Recovering the structural relationship (coherence relation) between two sentences of such a discourse involves a chain of inferences which results in resolving the ambiguities in the sentences. For two coherent sentences one is the expansion of the other. One is either describing an item in the earlier sentence or expanding on the idea centered around that item. The item being expanded upon is the focus of the sentence containing it. We called this item the Expandable item.

Interpreting the notion of coherence along with the notion of expansion led us to the principle of a minimal connectivity requirement between two sentences. According to the notion of minimal connectivity requirement, the lack of a relation between some item in a sentence and the Expandable item in another sentence is a signal of the lack of a coherence relation between the two sentences.

TEXT-ANALYZER is a mechanism to experiment with the theory stated above. The system's framework is to recover the connection between an incoming sentence and a sentence in the already interpreted portion of the discourse. Recovering the expandable item aids the system in the process of connecting another sentence to the sentence containing the expandable item. The information associated with the expandable item is an important part of the set of inferences that the system needs to make in order to establish a connection between the two sentences. The system uses the notion of minimal connectivity requirement to control its inferencing mechanism. The minimal connectivity requirement prevents the inferencer from wasteful attempts to connect two incoherent sentences.

TEXT-ANALYZER depends on MAINSTR, which serves as its knowledge base. MAINSTR is used as a source for producing an inferencing chain required for interpretation of a sentence. MAINSTR contains two kinds of information. First, there is the set of frame-like structures which represent the conceptual structure for the items encountered in the interpretation of the text. Second, a set of coherently connected event/states is accumulated through the analysis of different texts. This set is used by the system to recover a class of coherence relation which occurs between two sentences when the relation between the event/states described in the two sentences is either temporal or taxonomic. The system also realizes another type of coherence relation between two sentences (by means of the procedure **DETAIL**) which occurs between two sentences when one is describing an item in the other sentence.

The system was applied to three texts. The system interpretation of each text is demonstrated through the conceptual representation which it produces for the discourse of these sample texts. The representation is a graph which shows the interrelations between the sentences as well as among the items. The relations among the two sentences is the coherence relation between them; the relation among the items are reflected in connectivities among their conceptual structures.

5.2 Future work

TEXT-ANALYZER is just an experimental tool to test what we think is a reasonable way of analyzing a text. Certainly, to become a useful tool, TEXT-ANALYZER needs refinement almost in every aspect. There are factors (that we either overlooked or ignored to keep the system simple) whose consideration might lead us to a more general theory of text comprehension. In this section we propose some directions for future work which might further reveal the strengths and weaknesses of TEXT-ANALYZER.

The set of coherence relations used in any approach based on discourse coherence recognition is critical to the success of that approach. As mentioned earlier, computability and sufficiency are the main factors in selection of such a set. Careful analysis of more (and different kinds of) texts should lead us to a more certainly sufficient set of coherence relations.

With analysis of more texts, the knowledge base can be compiled of larger sets of coherently connected event/states and frame structures for the entities. This might improve the present status of the knowledge base and/or explore some factors that we failed to consider in the implementation of both the inferencer and knowledge base.

If understanding of a text is measured by the success of answering questions after interpreting it, then a question answering system should be an appropriate tool to test the usefulness of the representation that the system produces for a text.

We used the notions of expandability and minimal connectivity requirement to facilitate and control the inferencing mechanism. Refinements in both of these notions and/or discovering other means to reduce the complexity and search for the inferencer should have important effects in advancing discourse analysis toward a practically useful technology.

Appendix A.

MAINI

```
supermarket(X,frame(Y,isa(supermarket),
                      with([[employee,cashier],
                            [customer,customer],
                            [component,grocery],
                            [prop,money]]),
                      actions([[shop,[agt,customer],
                                     [obj,grocery],
                                     [instr,money],
                                     [loc,Y]]]))).

purse(X,frame(Y,isa(purse),
                with([[component,money]]),
                actions([]))).

bedroom(X,frame(Y,isa(bedroom),
                  with(Z),
                  actions([[sleep,[agt,person],[loc,Y]]]))) :-
                                                bedroom(W),
                                                addp(room,W,Z).

bedroom([[component,bed,pillow]]).

kitchen(X,frame(Y,isa(room),
                  with(Z),
                  actions([[eat,[loc,Y]],
                           [cook,[loc,Y]]]))) :-
                                    kitchen(W),
                                    addp(room,W,Z).

kitchen([[component,food,fridge,oven,counter,sink]]).

room(X,frame(Y,isa(room),
               with([[component,door,window]]),
               actions([]))).

grocery(X,frame(Y,isa(grocery),
                     with([[component,food,stationary]]),
                     actions([]))).

gun(X,frame(Y,isa(gun),
               with([]),
               actions([[rob,[agt,person],[instr,Y]]]))).

person(X,frame(Y,isa(person),
                  with([[sex,S]]),
                  actions([])) :-
                         sex(X,S).

animal(X,frame(Y,isa(animal),
                  with([]),
                  actions([]))).

business(X,frame(Y,isa(business),
                    with([[element,money]]),
                    actions([]))).

restaurant(X,frame(Y,isa(restaurant),
                      with([[employee,waitperson,cashier],
                            [customer,customer],
                            [furniture,table,light],
                            [prop,money,menu,food]]),
                      actions([[eat,[agt,customer],
                                    [obj,food],
                                    [loc,Y]]]))).

menu(X,frame(Y,isa(menu),
                with([]),
                actions([]))).

table(X,frame(Y,isa(table),
                 with([]),
                 actions([]))).

light(X,frame(Y,isa(light),
                 with([]),
                 actions([]))).

furniture(X,frame(Y,isa(furniture),
                     with([[elements,table,chair,light,couch]]),
                     actions([]))).

waitperson(X,frame(Y,isa(waitperson),
                      with(Z),
                      actions([[serve,[agt,Y],[obj,food],
                                      [recp,customer]]]))) :-
```

```
                                           addp(person,[],Z).

money(X,frame(Y,isa(money),
                with([]),
                actions([]))).

food(X,frame(Y,isa(food),
               with([]),
               actions([[eat,[obj,Y]]]))).

refrigerator(X,frame(Y,isa(refrigerator),
                        with([[component,food]]),
                       actions([[eat,[agt,person],[loc,kitchen],
                                                  [obj,food]]]))).
bank(X,frame(Y,isa(bank),
               with([[employee,clerk],
                     [customer,customer],
                     [component,money]]),
               actions([[deposit,[agt,customer],[obj,money],
                                  [loc,Y]],
                         [withdraw,[agt,customer],[obj,money],
                                   [loc,Y]]]))).

rocket(X,frame(Y,isa(rocket),
                 with([[weight,weight],
                       [fuel,fuel],
                       [type,type],
                       [loc,loc],
                       [prop,flare],
                       [observer,scientist]]),
                 actions([[flight,[ae,Y],[source,earth],
                                         [dest,eatrh]]]))).

plant(X,frame(Y,isa(plant),
                with([[parts,branch,root],
                      [type,type],
                      [seed,seed],
                      [hight,hight],
                      [fruit,fruit],
                      [loc,loc]]),
                actions([[grow,[ae,Y],[loc,earth]]]))).

seed(X,frame(Y,isa(seed),
               with([[parts,husk],
                     [shape,shape],
                     [type,type]]),
               actions([[distribute,[obj,Y]]]))).

husk(X,frame(Y,isa(husk),
               with([[type,type]]),
               actions([]))).

island(X,frame(Y,isa(island),
                 with([[component,plant],
                       [parts,shore],
                       [type,type]]),
                 actions([]))).

ocean(X,frame(Y,isa(ocean),
                with([[parts,wave,water]]),
                actions([]))).

radar(X,frame(Y,isa(radar),
                with([[prop,rocket]]),
                actions([[track,[obj,rocket],[instr,Y]]]))).


thing(X,frame(Y,isa(thing),
                with([]),
                actions([]))).

isa(scientist,person).
isa(waitpperson,person).
isa(cashier,person).
isa(customer,person).
isa(waiter,waitperson).
isa(waitress,waitperson).
isa(john,person).
isa(lobster,food).
isa(food,grocery).
isa(icecream,food).
isa(omlet,food).
isa(tip,money).
isa(fuel,thing).
isa(earthmound,thing).
isa(flare,thing).
isa(earth,thing).
isa(newmexico,thing).
isa(desert,thing).
isa(radar,thing).
isa(plane,thing).
isa(palm,plant).
isa(tree,plant).
isa(branch,thing).
isa(wave,thing).
isa(shore,thing).
```

Appendix B.

MAINR

```
prec([eat|X],[hungry|Y],X1,Y1,Cx,F)  :-
    check3(match([[agt,agt]]),arg1(X1,X),arg2(Y1,Y),
            changes([],Cx)).

prec([eat|X],[cook|Y],X1,Y1,Cx,F)  :-
    check3(match([[obj,obj]]),arg1(X1,X),arg2(Y1,Y),
            changes([],Cx)).

seq([eat|X],[pay|Y],X1,Y1,Cx,F)     :-
    check3(match([[agt,agt]]),arg1(X1,X2),arg2(Y1,Y2),
            changes([],Cx1)).
    check1(match([[agt,customer]]),arg(X2,X),changes(Cx1,Cx2)),
    check2(match([[obj,money]]),arg(Y2,Y),changes(Cx2,Cx)).

antec([eat|X],[travel|Y],X1,Y1,Cx,F)  :-
    check3(match([[agt,agt],[loc,dest]]),arg1(X1,X),arg2(Y1,Y),
            changes([],Cx)),
    check0(match([[loc,food]]),arg(X),F).

subseq([eat|X],[seat|Y],X1,Y1,Cx,F)  :-
    check3(match([[agt,ae]]),arg1(X1,X),arg2(Y1,Y),
            changes([],Cx)).

subseq([eat|X],[order|Y],X1,Y1,Cx,F)  :-
    check3(match([[agt,agt],[obj,obj]]),arg1(X1,X2),arg2(Y1,Y),
            changes([],Cx1)),
    check1(match([[loc,restaurant]]),arg(X2,X),changes(Cx1,Cx)).

subseq([eat|X],[serve|Y],X1,Y1,Cx,F)  :-
    check3(match([[obj,obj],[agt,benf]]),arg1(X1,X2),arg2(Y1,Y2),
            changes([],Cx1)),
    check1(match([[loc,restaurant]]),arg(X2,X),changes(Cx1,Cx2)),
    check2(match([[agt,waitperson]]),arg(Y2,Y),changes(Cx2,Cx)).

coord([eat|X],[like|Y],X1,Y1,Cx,F)  :-
    check3(match([[agt,agt],[obj,obj]]),arg1(X1,X2),arg2(Y1,Y),
              changes([],Cx1)),
         check1(match([[obj,food]]),arg(X2,X),changes(Cx1,Cx)).

prec([order|X],[travel|Y],X1,Y1,Cx,F) :-
         check3(match([[recp,agt]]),arg1(X1,X2),arg2(Y1,Y2),
              changes([],Cx1)),
         check1(match([[agt,waitperson]]),arg(Y2,Y3),changes(Cx1,Cx2)),
         check2(match([[dest,table]]),arg(Y3,Y),changes(Cx2,Cx)).

subc([travel|X],[come|Y],X1,Y1,Cx,F) :-
         check3(match([[agt,agt],[dest,dest]]),arg1(X1,X),arg2(Y1,Y),
              changes([],Cx)).

subseq([travel|X],[move_a|Y],X1,Y1,Cx,F) :-
         check3(match([[agt,ae],[dest,twd]]),arg1(X1,X),arg2(Y1,Y),
              changes([],Cx)).

subc([move_a|X],[go|Y],X1,Y1,Cx,F) :-
         check3(match([[ae,agt],[twd,dest]]),arg1(X1,X),arg2(Y1,Y),
              changes([],Cx)).

subseq([seat|X],[give|Y],X1,Y1,Cx,F) :-
         check3(match([[agt,agt],[ae,recp]]),arg1(X1,X),arg2(Y1,Y2),
              changes([],Cx1)),
         check1(match([[recp,customer],[obj,menu]]),arg(Y2,Y),
              changes(Cx1,Cx)).

conseq([give|X],[have_b|Y],X1,Y1,Cx,F) :-
         check3(match([[recp,agt],[obj,obj]]),arg1(X1,X),arg2(Y1,Y),
              changes([],Cx)).

seq([like|X],[tip|Y],X1,Y1,Cx,F) :-
         check3(match([[agt,agt]]),arg1(X1,X2),arg2(Y1,Y2),
              changes([],Cx1)),
         check1(match([[agt,customer]]),arg(X2,X),changes(Cx1,Cx2)),
         check2(match([[recp,waitperson]]),arg(Y2,Y),changes(Cx2,Cx)).

antec([pay|X],[have_a|Y],X1,Y1,Cx,F) :-
         check3(match([[agt,agt],[obj,obj]]),arg1(X2,X),arg2(Y1,Y),
              changes([],Cx1)),
         check1(match([[obj,money]]),arg(X2,X),changes(Cx1,Cx)).

antec([give|X],[have_a|Y],X1,Y1,Cx,F) :-
         check3(match([[agt,agt],[obj,obj]]),arg1(X1,X),arg2(Y1,Y),
              changes([],Cx)).

seq([eat|X],[depart|Y],X1,Y1,Cx,F) :-
         check3(match([[agt,agt],[loc,source]]),arg1(X1,X2),
              arg2(Y1,Y),changes([],Cx1)),
    check1(match([[loc,restaurant]]),arg(X2,X),
             changes(Cx1,Cx)).

subc([depart|X],[leave|Y],X1,Y1,Cx,F) :-
    check3(match([[agt,agt],[source,source]]),arg1(X1,X),
             arg2(Y1,Y),changes([],Cx)).

coord([grow|X],[appear|Y],X1,Y1,Cx,F) :-
    check3(match([[ae,ae],[loc,loc]]),arg1(X1,X2),arg2(Y1,Y),
             changes([],Cx1)),
    check1(match([[ae,plant]]),arg(X2,X),changes(Cx1,Cx)).

coord([grow|X],[produce|Y],X1,Y1,Cx,F) :-
    check3(match([[ae,ae]]),arg1(X1,X2),arg2(Y1,Y2),
             changes([],Cx1)),
    check1(match([[ae,plant]]),arg(X2,X),changes(Cx1,Cx2)),
    check1(match([[obj,seed]]),arg(Y2,Y),changes(Cx2,Cx)).

antec([grow|X],[sow|X],X1,Y1,Cx,F) :-
     check3(match([[ae,ae],[loc,loc]]),arg1(X1,X),arg2(Y1,Y2),
              changes([],Cx1)),
         check1(match([[ae,seed]]),arg(Y2,Y),changes(Cx1,Cx)).

prec([sow|X],[carry|Y],X1,Y1,Cx,F) :-
         check3(match([[ae,ae],[loc,dest]]),arg1(X1,X2),
                 arg2(Y1,Y),changes([],Cx1)),
         check1(match([[ae,seed]]),arg(X2,X),changes(Cx1,Cx)).

coord([distribute|X],[carry|Y],X1,Y1,Cx,F) :-
         check3(match([[ae,ae],[range,range],[instr,instr]]),
                 arg1(X1,X),arg2(Y1,Y),changes([],Cx)).

coord([distribute|X],[protect|Y],X1,Y1,Cx,F) :-
         check3(match([[ae,ae]]),arg1(X1,X),arg2(Y1,Y),
                 changes([],Cx)).

seq([produce|X],[distribute|Y],X1,Y1,Cx,F) :-
         check3(match([[obj,ae]]),arg1(X1,X),arg2(Y1,Y),
                 changes([],Cx)).

subseq([adapt|X],[protect|Y],X1,Y1,Cx,F) :-
         check3(match([[ae,ae],[to,from]]),arg1(X1,X),
                 arg2(Y1,Y),changes([],Cx)).

antec([takeoff|X],[ready|Y],X1,Y1,Cx,F) :-
         check3(match([[ae,ae]]),arg1(X1,X2),arg2(Y1,Y),
                 changes([],Cx1)),
         check1(match([[ae,rocket]]),arg(X2,X),changes(Cx1,Cx)).
```

```
subseq([flight|X],[takeoff|Y],X1,Y1,Cx,F) :-
        check3(match([[ae,ae],[source,source]]),arg1(X1,X),
               arg2(Y1,Y),changes([],Cx)).

subseq([takeoff|X],[fire|Y],X1,Y1,Cx,F) :-
        check3(match([[ae,ae]]),arg1(X1,X2),arg2(Y1,Y),
               changes([],Cx1)),
        check1(match([[ae,rocket]]),arg(X2,X),changes(Cx1,Cx)).

subseq([flight|X],[descend|Y],X1,Y1,Cx,F) :-
        check3(match([[ae,ae],[dest,dest]]),arg1(X1,X),
               arg2(Y1,Y),changes([],Cx)).

prec([takeoff|X],[stand|Y],X1,Y1,Cx,F) :-
        check3(match([[ae,ae],[source,loc]]),arg1(X1,X2),
               arg2(Y1,Y),changes([],Cx1)),
        check1(match([[ae,rocket]]),arg(X2,X),changes(Cx1,Cx)).

subseq([flight|X],[ascend|Y],X1,Y1,Cx,F) :-
        check3(match([[ae,ae],[source,source]]),arg1(X1,X),
               arg2(Y1,Y),changes([],Cx)).

antec([takeoff|X],[signal|Y],X1,Y1,Cx,F) :-
        check3(match([[ae,ae]]),arg1(X1,X2),arg2(Y1,Y),
               changes([],Cx1)),
        check1(match([[ae,rocket]]),arg(X2,X),changes(Cx1,Cx)).

subc([ascend|X],[rise|Y],X1,Y1,Cx,F) :-
        check3(match([[ae,ae],[source,source]]),arg1(X1,X2),
               arg2(Y1,Y),changes([],Cx1)),
        check1(match([[ae,rocket]]),arg(X2,X),changes(Cx1,Cx)).

subc([signal|X],[rise|Y],X1,Y1,Cx,F) :-
        check3(match([[instr,ae]]),arg1(X1,X),arg2(Y1,Y2),
               changes([],Cx1)),
        check1(match([[ae,flare]]),arg(Y2,Y),changes(Cx1,Cx)).

coord([rise|X],[protect|Y],X1,Y1,Cx,F) :-
        check3(match([[ae,from]]),arg1(X1,X2),arg2(Y1,Y2),
               changes([],Cx1)),
        check1(match([[ae,rocket]]),arg(X2,X),changes(Cx1,Cx2)),
        check1(match([[ae,person]]),arg(Y2,Y),changes(Cx2,Cx)).

subseq([protect|X],[withdraw|Y],X1,Y1,Cx,F) :-
      check3(match([[ae,agt]]),arg1(X1,X),arg2(Y1,Y),
             changes([],Cx)).

subc([descend|X],[plunge|Y],X1,Y1,Cx,F) :-
      check3(match([[ae,ae],[dest,loc]]),arg1(X1,X),
                 arg2(Y1,Y),changes([],Cx)).

coord([flight|X],[track|Y],X1,Y1,Cx,F) :-
          check3(match([[ae,obj]]),arg1(X1,X),arg2(Y1,Y),
                  changes([],Cx)).
```

Bibliography

[Alterman 82] Alterman, R.
A System of Seven Coherence Relations for Hierarchically Organizing Event Concepts in Text.
PhD thesis, Department of Computer Science, The University of Texas at Austin, 1982.

[Grosz 77] Grosz, B.
The representation and use of focus in dialog understanding.
Technical Report 151, Stanford Research Institue, 1977.

[Grosz 78] Grosz, B. and Hendrix, G.
A Computational Perspective on Indefinite Reference.
1978.
Unpublished technical report.

[Halliday 76] Halliday, M. and Hasan, R.
Cohesion in English.
Longman,London, 1976.

[Hirst 81] Hirst, G.
Discourse-Oriented Anaphora Resolution in Natural Language Understanding : A Review.
American Journal of Computational Linguistics 7(2):85-97, Apr-June, 1981.

[Hobbs 78] Hobbs, J.
Why Is discourse coherent?.
Technical Report 176, Stanford Research Institue, 1978.

[Lockman 78] Lockman, A. and Klappholz, D.
Contextual reference resolution in natural language processing.
Technical Report 70, Department of Computer Science, Rutgers University, 1978.

[Lockman 83] Lockman, A. and Klappholz, D.
The Control of Inferencing in Natural Language Understanding.
Computational Linguistics 5:59-70, January, 1983.

[Minsky 75] Minsky, M.
A framework for representing knowledge.
In Winston, P. (editor), *The Psychology of Computer Vision*, . McGraw-Hill, New York, 1975.

[Schank 77] Schank, R. and Abelson, R.
Scripts Plans Goals and Understanding.
Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1977.

[Schank 79] Schank, R.
Reminding and Memory Organization: An Introduction to MOPs.
Technical Report 170, Department of Computer Science, Yale University, 1979.

[Sidner 78] Sidner, C.
Levels of Complexity in Discourse for Anaphora Disambiguation and Speech Act Interpretation.
Proceeding of the fifth internationl joint conference on Artificial Intelligence 3:43-49, August, 1978.

[Sidner 83] Sidner, C.
Focusing and Discourse.
Discourse Process 6:107-130, Oct-Dec, 1983.

[Smith 81] Smith, M.
Knowledge-based Contextual Reference Resolution for Text Understanding.
PhD thesis, Department of Computer Science, The University of Texas at Austin, 1981.