2

This is to certify that the

thesis entitled

DISCOURSE REPRESENTATION THEORY AND PRONOUNS

presented by

Daehee Lee

has been accepted towards fulfillment
of the requirements for

M.S. degree in Computer Science

[illegible]
Major professor

Date 4-9-96

DISCOURSE REPRESENTATION THEORY AND PRONOUNS

by

Daehee Lee

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Computer Science

1996

ABSTRACT

DISCOURSE REPRESENTATION THEORY AND PRONOUNS

By

Daehee Lee

Pronominal Binding is one of the most complicated process of natural language understanding, since pronouns are differently interpreted, depending on which antecedent they refer to, and how they refer to their antecedents. This thesis resolves pronominal binding, tackling two problems: how a discourse must be represented to be accessible by computational procedures, and how computational procedures must be controlled in accessing the representations for antecedent search.

By designing a model in which syntax gives DRT some information about pronominal disjointness, and DRT searches for pronominal antecedents, this thesis can resolve many cases of referential, bound, and E-type pronominal binding, and pronominal binding in subordinations. This thesis also implements the theory in Prolog.

Dedicated to my mother, Sochool, my wife, Gemhee, and my three sons, Jaeyoung, Jaegook, and Jaesung

ACKNOWLEDGMENTS

I thank all the members of my committee, Carl Page, Barbara Abbott, George Stockman, and Betty Cheng, for their assistance for this study. Specially, I express my gratitude to Carl Page and Barbara Abbott.

Carl Page encouraged and helped me to be admitted to the graduate study of computer science, when I express my first interest in this field. His support and guidance were very helpful in my graduate study. He gave me good pieces of comments and information concerning this research.

I am grateful to Barbara Abbott for her encouragments and good comments on the linguistic part of this study, and for her guidance in my academic training in linguistics.

I express my deep and special gratitude to Alan Munn. Although he was not a member of my committee, he very happily gave his good comments on this project.

I am also indebted to OPHS (Office of Programs for Handicapper Students) and Tower Guard for their strong support. Without their reading service I could not have finished my graduate study at all. I give my special gratitude Michael Hudson, a vision specialist at OPHS, for his support and invaluable information on accessibility on campus.

I also thank my friends at computer science for their help and friendship. I give my special thank to Hyunduk John for many delightful conversations and good discussions throughout the courses of computer science.

I am also very truly grateful to my family. My mother, and brother, and sisters' strong support in pray and pride in me were good encouragments for me.

I never forget my wife's sacrificing support for me. I also thank all my three sons that they gave up their mother most of time when she helped me.

TABLE OF CONTENTS

INTRODUCTION

Typically, people communicate with computers by means of artificial languages. These languages are more suitable for computers rather than for people. In many cases, however, it will be more convenient if people can communicate with computers in natural languages. For example, to query some information from a large database, people should know to formulate the queries to retrieve the information which they need. If they can communicate with computers in natural languages, they can easily access the database and retrieve the information for their purposes.

For computers to be able to understand the natural languages, they must have the knowledge of the languages such as phrase structure rules, argument structures, some constraints on sentential derivations, some rules for semantic interpretations, and so on.

Relating to the knowledge about language use, the way in which pronouns are used and interpreted in a sentence or across sentences seems to be one of the most complicated processes in natural language understanding. Pronouns are semantically interpreted in some different manners, depending upon what they refer to, and how they are anaphorically linked with their antecedents. It is what we may call the problem of pronominal binding.

The goal of this thesis is to resolve intersentential pronominal binding in discourse, tackling two main problems: a representational problem and a computational problem. The former is related to the question of how discourse knowledge can be formally represented in a computational model, and the latter is related to the question of how to control the computational system to acces the representation of discourse for pronominal antecedent search: under which conditions anaphoric relations between different NP's and pronouns are possible. If we can understand and formalize the use of anaphoric pronouns it will contribute to enhancing natural language understanding systems.

To achieve the goal, this thesis will, first of all, classify the pronominal uses, and then investigate the representations and interpretations of anaphoric pronouns in a computational model of discourse--Discourse Representation Theory. The thesis will be organized as follows:

Chapter 1 will discuss Kamp's (1981) discourse representation theory (DRT) as a computational model in which the pronouns are represented and interpreted. Chapter 2 will classify and characterize pronouns on their use. Further, the representations and some constraints on pronominal interpretations will be in detail discussed in syntax and DRT.

Chapter 3 will discuss some issues of computing discourse representation structures in logic programming language. Chapter 4 will summarize the topic and discuss some issues for further research.

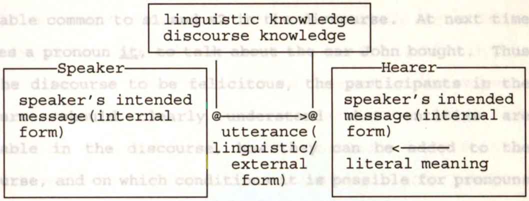
1. A Computational Model of Discourse

1.1 A General Model of Language Production and Comprehension

Language use (production and comprehension) is one of the human behaviors for communications. In discourse participants produce some linguistic forms to deliver their messages or mental states such as knowledge, thoughts, feelings, ETC. to share with others. For speakers to deliver messages, first of all, they build some internal forms or conceptual structures for the messages via some inference rules. Then they will produce some linguistic forms by means of some principles and rules. When they produce the linguistic forms for their intended meanings, they assume that their hearers can understand the inference rules and linguistic rules which they use for the production. The hearers now parse the linguistic forms, and translate them into some internal forms with the same linguistic rules that the speakers use for linguistic production. Then the hearers further apply the inference rules to the internal representations and understand the speakers' intended meanings.

The internal or conceptual forms will become a set of the common knowledge for the participants to share in discourse. As discourse continues to go further, the participants also update the common knowledge--adding some new information, resolving some references of anaphora, etc. To understand language use a very general computational model can be diagrammed in (1).

(1) (cf. Berwick (1983) Figure 2 p.29)


linguistic knowledge
discourse knowledge
Speaker
speaker's intended message(internal form)
@——>@
utterance(linguistic/ external form)
Hearer
speaker's intended message(internal form)
<——
literal meaning

From a computational point of view it is important to understand how participants construct their discourse knowledge. As one of the efforts to understand it, this thesis aims to enhance the representations of discourse knowledge, and the way to keep track of discourse information and compute the pronominal reference in discourse, given the representations.

1.2 Why Discourse Representations Are Necessary

In this section let us consider why the computational model requires some internal representations, and what factors are involved in the computation of discourse. Let us take some examples for concreteness. Consider that a discourse starts with the utterances in (2).

(2) s1: Do you know that John bought a car?

s2: Yes. It looks nice.

In (2) a speaker s1 introduces two entities, _John_ and _a car_, and specifies their relationship in which John is an owner of a car. With s1's utterance the two entities are available common to s1 and s2 in the discourse. At next time s2 uses a pronoun _it_, to talk about the car John bought. Thus for the discourse to be felicitous, the participants in the discourse should clearly understand what entities are available in the discourse, how they can be added to the discourse, and on which conditions it is possible for pronouns to be used to refer to previously mentioned entities. If participants do not know these factors, the discourse will fail to be felicitous.

According to Kamp (1981) and Heim (1983) a discourse entity is introduced into discourse by only some types of noun phrases such as proper nouns like _John_, deictic expressions like _that_ _this_ etc., indefinite NP's like _someone_ or _a car_. Other types of noun phrases such as definite NP's like _the car_, anaphoric pronouns like _he_, _it_, _her_, etc., and so on refer to the discourse referents already introduced in discourse. It reflects two computational procedures: the one introduces new entities; the other links one form to an already familiar entity.

Another importance is to capture an appropriate meaning of a sentence in discourse. The utterance of (3) potentially has two meanings, as described in (4).

(3) Two students ate a cake.

(4) a. There was one cake, and two students ate it together.

b. There were two students who each ate a cake.

If a hearer hears (3), he /shewe should be able to properly choose one of the two, (4.a) or (4.b), in the context. To do so the hearer should internally disambiguate the form of (3) as (4) at some representational level, and keep one of (4.a) or (4.b) in the knowledge base for the computational system to be able to correctly keep track of discourse interactions.

Let us consider another utterance of (5).

(5) A man entered the room. He looked like a detective.

Following Sidner (1983) and Grosz (1986), the computational system will be interested in updating the internal representations regarding what a discourse talks about, and when topic shift takes place. In (5) the discourse talks about the man who entered the room, and the pronoun <u>he</u> presumably refers to the man who just entered the room. If the computation catches what the discourse talks about, it will help the computation find what the pronoun refers to.

To represent discourse for intersential pronominal binding, this thesis will adopt and enhance Kamp's Discourse Representation Theory. In next section let us take this model into consideration for detail.

1.3 Discourse Representation Theory

Kamp (1981) proposes Discourse Representation Theory (DRT) in order to interface between a syntactic level and semantic interpretation. DRT assumes that there is a linguistic level between syntax and semantics, which is a level of Discourse Representation.

Let us consider the properties of DRT which make it possible for the computational system to search for pronominal antecedents across sentences in discourse. They will be described as follows: First, DRT is discourse-based. DRT constructs the representations of a discourse--what we may call discourse representation structures (DRS's). A single DRS accumulates all discourse information. Second, DRT can explicitly specify the intersentential scope and anaphora-antecedent relations across sentences such as modal subordinations, quantificational subordinations, and pronominal references, providing a unified representation of a discourse structure which contains all intersentential relations in a discourse. Third, DRT is independent of a theory of syntax, since the level of DRT is placed after syntactic analyses. So any syntactic analysis is suitable for determining meaning. And fourth, DRT contains a theory of truth-conditions. Truth is defined in terms of embedding DRS's in a model.

The first and second property make it possible for DRT to be adequate for solving the intersentential pronominal binding. The third property allows syntax to help DRT to search for pronominal antecedents in a DRS, providing some constraints on anaphora-antecedent relations.

At the level of DRT a DRS consists of a pair of two sets, <U, C> where U is a set of discourse referents, and C a set of properties and relations, including negation, disjunction, implication, etc. The former is called a universe and the latter a condition (or a predication) on a universe.

A DRS initially consists of a pair of empty sets, <U,C>, since there is no information available in the discourse. To start some conversation, the participants should use some linguistic expressions which can introduce some discourse referents and their properties or relations into a DRS. As discourse goes on, however, a sequence of DRS's will be constructed or revised by some algorithms.

According to Kamp (1981) and Heim (1983) a discourse referent is introduced into a DRS by some types of NP's only. That is, proper nouns, deictic expressions, indefinite NP's introduce new referents into a DRS, and other types of NP's such as definite NP's, anaphoric pronouns, and so on refer to the discourse referents already introduced in a DRS.

Let us take some examples for DRT now. Consider (6).

(6) John saw Mary.

In (6) a discourse referent, u, is introduced into a DRS for <u>John</u>, and another discourse referent, v, is added into a universe, U, for Mary. The relation between John and Mary, saw(u,v), is added into a condition, C, which explicitly constrains the relationship between two discourse referents, u and v. Thus (6) is linearly represented in (7a) and schematically represented in (7b).

(7) a. D1 = <{u,v}, {John(u), Mary(v), saw(u,v)}>.

b.

```
┌──────────────┐
│ u v          │
│ John(u)      │
│ Mary(v)      │
│ saw(u, v)    │
└──────────────┘
```

After (7) is constructed the conversation is assumed to go further with (8). Then we may have (9) as a DRS.

(8) John owns a donkey, and Mary owns a cat.

(9)

```
┌──────────────┐
│ u v w x      │
│ John(u)      │
│ Mary(v)      │
│ saw(u, v)    │
│ donkey(w)    │
│ cat(x)       │
│ own(u, w)    │
│ own(v, x)    │
└──────────────┘
```

A DRS is a discourse-based structure. So several subsequent sentences can be accumlatively accommodated in one DRS. Furthermore, even a sentence can have more than one DRS if necessary, as described just below.

Kamp (1981) suggests that a DRS may be embedded in another DRS. Let us take (10) for an example.

(10) a. John does not own a car.

b. not [a x: car x] (John own x). (preferred reading)

c. [a x: car x]not(John own x). (nonpreferred reading)

(11)

```
+----------------+
| u   v          |
| John(u)        |
| car(v)         |
| ¬ own(u, v)    |
+----------------+
```

If we represent (10.a) as (11) in the same way as we have done for (8), the DRS, (11), corresponds to (10.c), whose reading is not preferred for some reason.

That is, the truth of (10.c) is determined by two factors: the one is whether a unique car exists in a model, and the other is whether there is a relation of John owning that car. So (11) may be false even if there is no specific car in a model, regardless of whether John owns a car or not.

If a speaker of (10.a) intends to mean (10.b), however, the representation of (11) is problematic.

To solve the problem with the representation for (10.b), we may represent it as (12). In comparison with (11) the DRS, (12), corresponds to (10.b).

(12)

```
+-----------------------+
|  u                    |
|  John(u)              |
|                       |
|     +--------------+  |
|     |  v           |  |
|  ¬  |  car(v)      |  |
|     | own(u, v)    |  |
|     +--------------+  |
+-----------------------+
```

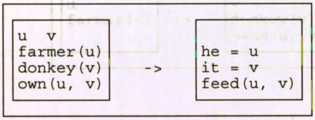
In (12) Kamp (1981) calls the outer box or DRS a superordinate DRS to the inner box or DRS embedded in it, (henceforth sup-DRS), and the embedded DRS a DRS subordinate to the outer DRS (henceforth sub-DRS). A sub-DRS is also used for if-then constructions, universal quantifiers, pragmatic accommodations, modal/quantificational subordinations[1], etc.

One important thing is that for anaphor-antecedent relations the discourse referents in sup-DRS are accessible to the sub-DRS, but not vice versa.

Let us take an example for how a sub-DRS works for if-then constructions and universal quantifiers. (13) is an if-then construction, and can be represented in (14).

(13) If a farmer$_i$ owns a donkey$_j$, then he$_i$ feeds it$_j$.

(14)



In (14) the leftside box is a DRS superordinate to the rightside box, and conversely, the rightside box is a DRS subordinate to the leftside box.

Following Kamp (1981), a universal quantifier like <u>every</u> should be represented by an if-then construction in DRT. Consider (15).

(15) a. Every farmer owns a donkey.

[1]I will discuss pragmatic accommodations with respect to E-type pronouns in detail in chapter 2.

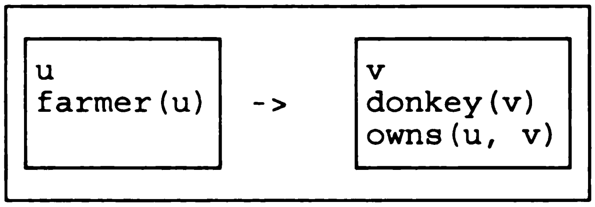
b. [every x: farmer x] [a y: donkey y] (x owns y).
(preferred reading)

c. [a y: donkey y] [every x: farmer x] (x owns y).
(nonpreferred reading)

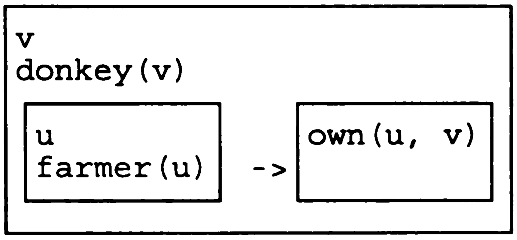
Kamp (1981) and others have discussed a DRS, (16), for (15.a), but it corresponds to (15.b) only. Since the sentence (15.a) is ambiguous as (15.b) and (15.c), although (15.c) is not preferred for some reason here, DRT should be able to represent (15.c) as a DRS at the level of DRT.

So (15.c) can be represented as (17) in comparison with (16).

(16)



(17)



The difference in the interpretation between (11) and (12), and between (16) and (17) will be much more clear after we review the interpretation of a DRS below.

If a DRS is constructed by some rules, then it is used to determine the truth of a discourse. According to Kamp (1981, P. 278):

> "A sentence S, or discourse D, with representation m is true in a model M if and only if M is compatible with m; and compatibility of M with m, .., can be defined as the existence of a proper embedding of m into M, where a proper embedding is a map from the universe of m into that of M which, .., preserves all the properties and relations which m specifies of the elements of its domain."

For this let us consider the truth conditions in the model-theoretic semantics first. In order for a sentence S to be true in Model M, there is a mapping which maps the individuals in S into a subset of individuals in M and which maps the properties or relations of the individuals in S into a subset of properties or relations in M. In the same way a DRS D is true in a model M iff there is such a mapping that the set of discourse referents in the DRS is mapped into a subset of the individuals in a model M, and that the properties or relations in the DRS are mapped into the subset of the properties or relations in M.

Then let us evaluate the truth of the DRS, (7). The DRS, (7), with a set of discourse referents, $U_D=\{u,v\}$, and a set of properties and relations, $C_D=\{\text{John}(u), \text{Mary}(v), \text{saw}(u,v)\}$, is true in a model M with a set of individuals, U_M, and an assignment function, G_M, iff $<G_M(\text{John}), G_M(\text{Mary})>$ is a member of $G_M(\text{Saw})$ and there is a mapping g from U_D to U_M such that $g(u)=G_M$ (John) and $g(v)=G_M(\text{Mary})$ and $<g(u), g(v)>$ is a member of $G_M(\text{Saw})$.

Consider the case of (9). Let us partially evaluate <u>John owns a donkey</u> in (9) for simplicity because other parts can be evaluated in the same way. The truth value is determined with such an interpretation that the DRS, (9), which partially consists of $<U_D, C_D> = <\{u,w\}, \{John(u), donkey(w), own(u,w)\}>$, is true in a model M with a set of individuals, U_M, and an assignment function, G_M, iff there is any individual x in U_M such that $G_M(x)$ belongs to $G_M(donkey)$, and $<G_M(John), G_M(x)>$ is a member of $G_M(Own)$ and there is a mapping g from U_D to U_M such that $g(u)=G_M(John)$ and $g(w)=G_M(x)$ and $<g(u), g(w)>$ is a member of $G_M(Own)$.

The DRS, (12), can be interpreted in a recursive way. The DRS, (12), consists of $D = <U_D, C_D> = <\{u\}, \{John(u), not(D_1)\}>$ where $D_1 = <U_{D1},C_{D1}> = <\{v\},\{car(v),own(u,v)\}>$. It is true in a model M with a set of individuals, U_M, and an assignment function, G_M, such that there is a mapping g from U_D to U_M such that $g(u)=G_M(John)$ and D_1 is false (that is, $not(D_1)$ is true). Now D_1 is interpreted in the same way as (9).

The difference between (11) and (12) is that car(v) should be true (or there should be a specific car) for (11) to be true, and that car(v) does not need to be true for (12) to be true. So this simulates the nonspecific use of an indefinite NP.

Kamp (1981) suggests that if-then sentences and universal quantifiers should be treated in the same way for interpretation. For example, the statement, if A then B, is true iff B should be true in all situations in which A is true.

In next chapter let us look at pronominal representation and their interpretation in DRT.

2. Pronominal References and DRT

First of all, let us start the discussion about the pronominal uses, taking a brief look at Evans's pronominal classification because he not only introduced a new pronominal use, but also reflected the prior pronominal analyses well.

Evans (1980) proposes that the pronouns be classified into four types, depending upon their occurrences, as follows:

(1) (i) "Pronouns used to make a reference to an object (or objects) present in the shared perceptual environment, or rendered salient in some other way." (p.337)

(ii) "Pronouns intended to be understood as being coreferential with a referring expression occurring elsewhere in the sentence." (p.337)

(iii) "Pronouns which have quantifier expressions as antecedents, and are used in such a way as to be strictly analogous to the bound variables of the logician." (p.337)

(iv) Pronouns which also "have quantifier expressions as antecedents," but "are not bound by those quantifiers." (p.338)

Let us take a close look at the pronominal properties and references, based on the classification of (1) one by one below.

2.1 Deictic vs. Anaphoric Pronouns

In the case of (1.i), first of all, a pronoun is used as a referring expression as in (2). The sentence (2.a) may be uttered with some gestural or perceptual demonstration to a student coming into the room. Even without any gestural or perceptual demonstrations to an object or objects, a pronoun may be used as a referential expression in the context in which an object or objects are salient, and a pronoun refers to them. For example, suppose a boy runs back and forth and makes noise in a public room for a while and then leaves the room. Right after that (2.b) is uttered.

(2) a. He is a smart student.

b. He is a spoiled boy.

Such pronouns can be called deictic pronouns. The deictic pronouns are semantically interpreted as a referring expression as the pronouns classified as (1.ii) are, but they are different from (1.ii) in the uses and representations.

Let us observe the use of the pronouns of (1.ii) first, and then discuss (1.i) and (1.ii) comparatively.

Now the (1.ii)-type pronouns can be exemplified in (3).

(3) a. When $John_i$ had finished, he_i left.

b. $John_i$ was strong, and he_i broke in the door.

c. $John_i$ loves his_i mother.

d. $John_i$ thinks that he_i is sick.

In case one expression determines the semantic value of the other expression, the former should be a constituent of the latter. In (3), however, the semantic value of John determines the semantic value of his or he, even though John is not a constituent of his or he. In this case it is said that the pronoun his or he is used as an anaphor on John, and that John is an antecedent of his or he. The pronoun his or he is interpreted as referential as in (2). That is, the anaphoric pronoun his or he refers to the same object that the antecedent John refers to. Neale (1990) calls the (1.ii)-type of a pronoun an anaphor as a referential usage.

At the level of DRT we can distinguish deictic pronouns from anaphoric pronouns in some different way. Let us consider a model of discourse first. A model of discourse consists of discourse referents and their properties or relations. While a conversation goes on, the discourse referents and their properties in the model continue to be updated in such a way that new objects or properties are introduced into the model, or the discourse referents are reordered with respect to the times when they are referred to. Furthermore, the manner in which objects and/or properties are added to the model is to express them linguistically only. Although some objects are rendered salient by nonlinguistic factors such as perceptual or gestural ones, they cannot be discourse referents in the model of discourse if they are not expressed linguistically. I will assume that this is a constraint on the introduction of discourse referents into a discourse model.

The terms "object" and "discourse referent" need to be made clear. The term "object" refers to the "entities that belong to the real world" (Kamp 1981 p.283), and the term "discourse referent" refers to the entities that belong to the representation of a discourse, which denote the objects in the real world or a more abstract device for intermediate maintenance of anaphoric relations.

Then let us assume that the anaphoric relation indicates the configurational relation between an anaphoric expression and a discourse referent. That is, only a discourse referent can be a potential antecedent which an anaphor refers to, and only the discourse referents at sup-DRS's or in the same DRS are accessible to an anaphor.

The pronouns of (1.i) refer to objects and the pronouns of (1.ii) refer to discourse referents. More generally speaking, (1.i) is distinguished from (1.ii, iii and iv) in such a way that (1.i) applies to the pronouns which refer to objects out of a discourse model, and introduce some discourse referents into a discourse model, while (1.ii, iii and iv) refer to a discourse referent or referents already existing in the discourse model. The former may be called deictic pronouns, and the latter anaphoric pronouns.

Then we can define deictic pronouns in DRT as follows:

(4) Pronouns are deictic if they are used to make reference to an object or objects which are out of a

discourse model or rendered salient in just a nonlinguistic way.

We can also define anaphoric pronouns as follows:

(5) Pronouns are anaphoric if they are linked with discourse referents in a DRS.

With respect to (4) and (5), the pronouns in (2) are deictic, and the ones in (3) are anaphoric.

Kamp (1981 P.282-283) assumes without any specific explanation that deictic and anaphoric pronouns be treated under the same principle:

> "...the mechanisms which govern deictic and anaphoric occurrences of pronouns are basically the same. ... both deictic and anaphoric pronouns select their referents from certain sets of antecedently available entities."

It would be wrong if the deictic and anaphoric uses of pronouns are treated in the same way just for the reason that both "select their referents from certain sets of antecedently available entities." (p.283)

Apparently, the mechanisms by which deictic and anaphoric pronouns find their referents are different:

First, the properties of sets from which the two pronouns select are different. As I have mentioned just above, the entities which the deictic pronouns refer to are in the real world; that is, they directly refer to the entities in the real world. In contrast, the anaphoric pronouns refer to the entities in the representation of a discourse, and indirectly refer to the entities in the real world via the discourse referents.

Second, deictic pronouns refer to their objects in a nonlinguistic way -- perceptual or gestural demonstration -- but anaphoric ones determine their antecedents in very complex ways--grammatical agreements, syntactic/DRS configuration (or maybe thematic roles), a selected topic, and so on.

Third, deictic pronouns introduce new discourse referents to a discourse representation, but anaphoric ones can never introduce new discourse referents, and instead reorder the referents in the discourse.

So, instead of the unified treatment of deictic and anaphoric pronouns, the distinction between deictic and anaphoric pronouns is necessary to establish the strategies to determine their referents properly.

This classification of pronouns in DRT will solve some problem with Evans's classification. With (1.ii) Evans suggests that coreferentiality takes place in the same sentence. But intuitively, (6) is semantically the same as (3.b).

(6) John$_i$ was strong. He$_i$ broke in the door.

Following Evans (1980), the pronoun he in (6) is no more a (1.ii)-type pronoun, because the latter is not in the same sentence as the former and violates (1.ii.). Evans may answer this question in such a way that the former sentence makes John salient in a context and he in the latter refers to the salient object. So the pronoun in (6) can be treated as a

(1.i)-type. But it is not adequate if (3.b) and (6) are differently treated, although they seem to be the same.

In addition to this inconsistency, we may say that the pronoun in (3.b) can also be classified as (1.i) because the prior clause makes John salient. Hence the case of (3.b) may be classified as either (1.i) or (1.ii).

However, our analysis treats (3.b) and (6) as anaphoric pronouns, since the pronouns refer to discourse referents in a DRS.

Now let us consider the DRS's of (2) and (3). Kamp (1981) suggests that a proper noun should be placed at the top-level of a DRS. Here the top level of a DRS indicates the highest superordinate level. He did not discuss this rule, but the reason that the a discourse referent for proper nouns are introduced at the top-level of a DRS seems to make the referent globally accessible to the anaphora. Then a deictic pronoun can be treated in the same way as a proper noun. For deictic pronouns, (2.a) may be represented as (7), and for anaphoric pronouns, (3.b) will be represented as (8).

(7)

```
u
he(u)
smart (u)
student (u)
```

(8)

```
v
John(u)
strong (u)
door (v)
he=u
break_in(u, v)
```

2.2 Interactions between Syntax and DRT

2.2.1 A General Constraint in Syntax

Before discussing how DRT searches for pronominal antecedents, we should take the role of syntax into considerations.

Syntax provides pronominal contra-indices between pronouns and some noun phrases for DRT. It means that some pronouns should not be coindexed with some noun phrases with the same indices that the pronouns have in syntax. If syntax specifies some contra-indices between pronouns and noun phrases, then DRT will exclude the NP's for pronominal antecedent search.

For example, assume that syntax should specify a contra-index between <u>he</u> and <u>John</u> in (9.a). Then DRT will allow a DRS like (9.b) in which <u>he</u> and <u>John</u> are anaphorically linked, while it excludes (9.c) which indicates that <u>he</u> refers to

<u>John</u>.

(9) a. He$_{*i}$ likes John$_i$.

b.

u
John$_i$ (u)
he$_{*i}$ (v)
likes (v, u)

c.

u
John$_i$ (u)
he$_{*i}$ = u
likes (u, u)

How can syntax provide such contra-indices between pronouns and NP's, then? We assume that syntax should parse a sentence with some constraints, and that the Principle C of the binding theory in (10) be one of such constraints.

(10) Principle C of the Binding Theory

A R-expression[2] with an index should not be c-commanded by another expression with the same index in syntax.

(11) X c-commands y iff all the categories which dominate x also dominate y[3].

For example, in (12) the pronoun <u>he</u> with an index i c-commands <u>John</u> with the same index i so that it violates the Principle C. Thus Syntax should provide a contra-index between <u>he</u> and <u>John</u>.

(12) a. He$_{*i}$ thinks that John$_i$ is rich.

[2]Roughly speaking, R-expressions are the ones excluding anaphoric categories: pronouns, reflexives, and reciprocals.

[3]The c-command relationship is related to pure syntactic configurations within a sentence.

```
b.     S
      / \
     NP  VP
     |  /  \
    he V    S'
       |   / \
   thinks C   S
          |  /  \
       that NP   VP
             |    |
          John [is rich]
```

On the other hand, in (13) he does not c-command John at all, and syntax does not provide any contra-index between he and John.

(13) a. John$_i$ thinks that he$_i$ is rich.

```
b.     S
      / \
     NP  VP
     |  /  \
  John V    S'
        |  / \
   thinks C   S
          |  /  \
       that NP   VP
            |    |
           he [is rich]
```

This syntactic constraint and interaction with DRT will treat pronouns in a consistent way.

For example, a question about Evans's treatment of (3) has been raised by Bosch (1983). Following Bosch (1983), the unified treatment of all the pronouns in (3) is irrelevant. He proposes that the pronouns in (3.a.-b.) and the ones in (3.c.-d.) should be treated differently, since the pronouns in (3.a.-b.) are pragmatically linked to their antecedents, and the pronouns in (3.c.-d.) are syntactically linked to their antecedents. Furthermore, Bosch argues that the syntactically functioning pronouns are semantically interpreted as nonreferential expressions; pragmatically functioning ones are interpreted as referential expressions like proper names.

To support his argument, he suggests the following as an evidence under the assumption that if an expression is referential then it is replaceable by another referring expression:

(14) a. Fred$_i$ thinks he$_i$ is sick. (p.41)

b. John$_i$ looks pale, and Fred thinks he$_i$ is sick.

(15) a. ?Fred$_i$ thinks the old malinger$_i$ is sick.(p.42)

b. John$_i$ looks pale and Fred thinks the old malinger$_i$ is sick.

Replacing the pronouns in (14) with a referring expression shows us a difference between (14.a) and (14.b). Following Bosch, the reason that (15.b) is grammatical is that the pronoun in (14.b) is referential and so replaceable with another referring expression; the pronoun in (14.a) is not referential and cannot be replaceable with another referring expression.

Bosch argues that in (14.a) the pronoun <u>he</u> is syntactically linked to its antecedent, depending on the syntactic agreements, regardless of whether or not the antecedent is a referring expression. The pronoun <u>he</u> in (14.b) as a referring expression, however, requires another referring expression as its antecedent. They refer to two independently referring expressions which have the same referent, which may be called coreferential. However, the pronoun in (14.a) does not require a referring expression as its antecedent because it functions as a nonreferential expression. So Bosch proposes that, since syntactically functioning pronouns and pragmatically functioning pronouns have their own mechanism for linking to their antecedents, the pronouns in (3.a.-b.) and the ones in (3.c.-d.) be treated under different principles.

However, the ungrammaticality of (15.a) does not depend upon whether or not the pronoun in (15.a) is referential, as Bosch analyzes. Rather, (15.a) violates the condition (c) of the binding theory specified in (10). To put it in other words, the phrase <u>the old malingerer</u> in (15.a) is coindexed with <u>Fred</u> which c-commands it. On the other hand, the phrase <u>the old malinger</u> in (15.b) is coindexed with <u>John</u> but not c-commanded by it. Thus the ungrammaticality of (15.a) seems to be attributed not to the characteristics of pronouns but to the characteristics of the r-expression.

2.2.2 More Constraints in Syntax

To consider more constraints on pronominal antecedency, we should classify pronouns into three types in syntax: referential, bound, and E-type pronouns. Referential pronouns which refer to referential expressions do not require any other constraint but the Principle C described in the previous section. However, syntax should provide some other constraints on the second and third types of pronouns. In this section let us consider the characteristics of these pronouns and their constraints.

2.2.2.1 Bound Pronouns

Now let us take (1.iii) into consideration. An anaphoric pronoun can also be interpreted as a bound variable if it is used anaphoric on quantifiers and bound by those quantifiers:

(16) a. Every boy_i thinks he_i is smart.

b. Some boy_i thinks that he_i is smart.

c. No boy_i thinks that he_i loves Mary.

In (16) the pronouns <u>he</u> are interpreted as a variable bound by the quantifiers <u>every</u>, <u>some</u> and <u>no</u> in (17), respectively. The pronouns in (16) will belong to (1.iii)-type pronouns. Neale (1990) also refers to (1.iii)-type of a pronoun as an anaphor as a bound variable.

(17) a. [every x: boy x] (x thinks that x is smart)

b. [some x: boy x] (x thinks that x is smart)

c. [no x: boy x] (x thinks x loves Mary)

With respect to the pronouns as bound variables, Evans (1980), Reinhart (1983), May (1985) and others suggest the syntactic constraint that a pronoun anaphoric on a quantifier is bound by that quantifier iff that quantifier c-commands that pronoun. Furthermore, May (1985) has shown that the syntactic level is the level in which the configurational constraint of c-command should take place.

May (1985) proposes two principles for quantifier scope and the pronouns as bound variables:

(18) a. The scope of x is the set of nodes that x

c-commands at LF. (p.5)

b. A pronoun is a bound variable only if it is within the scope of a coindexed quantifier. (p.21)

The principle of (18) explains why the pronouns in (16) can be interpreted as bound variables.

This constraint is not applicable to referential pronouns. Take the following example.

(19) a. His$_i$ mother loves John$_i$.

b.* His$_i$ mother loves every boy$_i$.

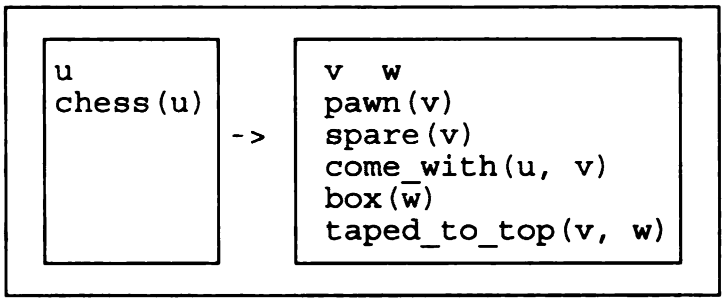
In (19.b) the pronoun _his_ cannot be coindexed with _every boy_, since it violates (18). However, (19.a) the pronoun _he_ can refer to _John_. So syntax should provide a contra-index between _his_ and _every boy_ for DRT.

While syntax simply specifies contra-indices, DRT will resolve pronominal binding. Let us consider (20).

(20) Every chess set comes with a spare pawn. It is taped to the top of the box. (Kadmon P. 292)

In (20), on the preferred reading, the phrase _every chess_ takes a wide scope over the phrase _a pawn_, and also a wide scope over the pronoun _it_ if the pronoun _it_ is coindexed with the phrase _a pawn_. If the first sentence in (20) is not processed through DRT, it may be difficult to properly interpret the second sentence containing the pronoun under the scope of the prior sentence. The sentence of (20) can, however, be represented as (21) if Roberts's (1989) discourse subordination applies for (20) here. It is a case of quantificational subordination.

(21)



2.2.2.2 E-type Pronouns

Here are more complex examples of the anaphoric pronouns which are not interpreted straightforwardly, however:

(22) a. John bought some donkeys$_i$ and Bill vaccinated them$_i$.

b. [some x: donkey x](John bought x and Bill vaccinated x)

c. [some x: donkey x](John bought x) & [the y: donkey y & John bought y](Bill vaccinated y)

In (22.a) the pronoun <u>them</u> and the quantifier phrase <u>some donkeys</u> seem to be anaphorically related, but the phrase <u>some donkeys</u> does not c-command the pronoun <u>them</u>. Hence the pronoun <u>them</u> cannot be bound by the quantifier as defined in (18).

Furthermore, the pronoun in (22.a) cannot, actually, be interpreted as a bound variable, either. For example, if we translate (22.a) to a logical form as a bound variable interpretation of them, then (22.a) will be logically represented as (22.b). As Evans (1980) points out, (22.b) can be true even in case John bought 10 donkeys and Bill vaccinated 9 donkeys. So the pronoun them in (22.a) should not be interpreted as the case of bound variable like (22.b).

Rather, following Evans (1980), naturally, (22.a) is interpreted in such a way that John bought some donkeys and Bill vaccinated all the donkeys that John bought. The pronouns which are anaphoric on the quantifiers but not bound by those quantifiers are a (1.iv) or E-type pronoun.

An E-type pronoun is also applicable to an indefinite singular phrase. In this case it can have two different readings: either maximal or existential, depending on the context in which it occurs. Consider Heim's (1982) example:

(23) Every man who owned a slave$_i$ owned its$_i$ offspring.

In this sentence the indefinite a slave does not c-command the pronoun its, but can be co-indexed with it. Thus the pronoun should be understood as an E-type pronoun. This sentence will read a maximal reading like (22.c) so that if a man owned three slaves then he would own all the offsprings of all the slaves that he owned.

According to Evans, an E-type pronoun and its indefinite antecedent should be uniquely existential:

(24) John bought a donkey$_i$ and Harry vaccinated it$_i$.

That is, in (24) there must be at most one donkey and at least one donkey that John bought, and Harry vaccinated that donkey. Such uniqueness cannot be carried out by the existential quantifiers themselves; the E-type pronoun makes the indefinite antecedent unique.

If Evans's analysis is true, then the E-type pronouns are very similar to the Russell's analysis of definite descriptions. Following Russell's analysis of definite descriptions, the singular definite descriptions show uniqueness. For example, Russell (1919) analyzes the sentence <u>THe author of Waverley was Scotch.</u> as follows: (p.218)

(25) a. <u>x wrote Waverley</u> is not always false.

b. <u>if x and y wrote Waverley, x and y are identical</u> is always true.

c. <u>if x wrote Waverley, x was Scotch</u> is always true.

If E-type pronouns have similar properties to definite descriptions, can we treat E-type pronouns like definite descriptions?

Evans argues that E-type pronouns have their own "referents fixed by descriptions" like proper names. We may call this analysis E-type pronouns as rigid designators[4].

Evans argues that it is necessary to analyze the E-type pronouns as rigid designators, since the E-type pronouns

---

[4]This term is from Kripke (1972). An expression is a rigid designator if it makes reference to the same object in every possible world in which it exists, although it does not necessarily exist in all the possible worlds (Kripke 1972, p.111). Kripke argues that proper names are rigid designators, and the definite descriptions are nonrigid designator. The examples are "Nixon" and "the president of the United States".

cannot be used to refer to an empty set, and do not show the scope ambiguity over sentential operators that overt descriptions do. Let us take the following:

(26) *John bought no donkey$_i$, and Harry vaccinated it$_i$.

(27) a. A man$_i$ murdered Smith, but John does not believe that he$_i$ murdered Smith.

b [the x: man x & x murdered Smith] (John does not believe that (x murdered Smith))

c. *John does not believe that ([the x: man x & x murdered Smith] (x murdered Smith)).

The E-type pronouns should have their references fixed by description, but it fails in (26). Furthermore, the E-type pronoun <u>he</u> in (27) has de re reading but no de dicto reading. If it were definite description, then it should also show the de dicto reading.

Now let us discuss the representation of E-type pronouns in DRT.

Kamp (1981), first of all, suggests that (28) be represented as (29).

(28) Every Farmer who owns a donkey$_i$ beats it$_i$.

(29)

$$\boxed{\boxed{\begin{array}{l} u \quad v \\ \text{farmer}(u) \\ \text{donkey}(v) \\ \text{own}(u,\ v) \end{array}} \quad \text{->} \quad \boxed{\begin{array}{l} \text{it} = v \\ \text{beat}(u,\ v) \end{array}}}$$

Then the DRS (29) with a universe U_D and condition C_D is interpreted in such a way that (29) is true in a model M with a set of individuals U_M and an assignment function G_M iff there is a mapping function g from D to M such that for all g in which g(u) is a member of G_M(farmer) and g(v) a member of G_M(donkey) and <g(u) g(v)> belongs to G_M(Own), <g(u) g(v)> belongs G_M(beat). In other words every pair of _farmer_ and _donkey_ is truth-conditionally evaluated with respect to the relations of _beat_ and _own_. This representation demonstrates Geach's analysis of (29).

Geach (1968) argues that the pronoun _it_ in (28) must be regarded as a bound variable, bound by _a donkey_, which is given wide scope. So he suggests that (30.a) should be a logical form of (28).

(30) a. $\text{Every}_{x,y}$ [x is a farmer ^ y is a donkey ^ x has y] [x beats y].

b. every_x[x is a farmer ^ some_y[y is a donkey ^ x has y]] [x beats y].

If (30.b) is treated as the logical form of (28), it might be hard to analyze the pronoun _it_ in (30.b), because _a donkey_ cannot C-command it. So, following Neale (1990), (30.a) can be derived from (30.b) as follows: Since (some x p) -> q and [every x] (p -> q) are logically equivalent where q contains no free occurrence of x, the form (30.b) is logically equivalent to (30.a). In this way, the indefinite NP a donkey gets universal force.

Although this DRS represents a maximal/universal reading of an E-type pronouns, this analysis does not reflect an existential reading of E-type pronouns, as described below.

The first case is that the universal reading of a donkey sentence will give rise to a problem if we replace every with most. Let us take Chierchia's (1992) example: (P. 119)

(31) a. Most farmers that have a $donkey_i$ beat it_i.

b. $Most_{x,y}$ [farmer(x) ^ donkey (y) ^ x owns y] [x beats y]

c. $Most_x$ [farmer(x) ^ [a y [donkey (y) ^ x owns y]]][a y[x beats y]]

If we interpret (31.a) in such a way that the quantifier most has scope over the pair of farmer and donkey as in (31.b), then it will give us a wrong truth condition in some situations. For example, let us consider the situation in which there are five farmers such that one farmer has 99 donkeys and each of the other four farmers has one donkey, and that only the farmer who has 99 donkeys beats his donkeys. In this situation (31.b) is true. However, (31.a) is naturally interpreted as false in that situation. In order for (31.a) to be true, 3 or 4 of five farmers should beat their own donkeys regardless of the number of donkeys which farmers have. In this sense the quantifier most in (31.a) should be quantified over farmers only and not over donkeys as in (31.c). In the same sense we need another representation in which the quantifier every in (28) is interpreted as a quantification over farmer only.

The second problem of (29) is that the representation of (29) cannot express that the indefinite NP, a donkey, has the existential force in Russell's term. It will be more clear if we again look at the example which Chierchia takes: (P. 115)

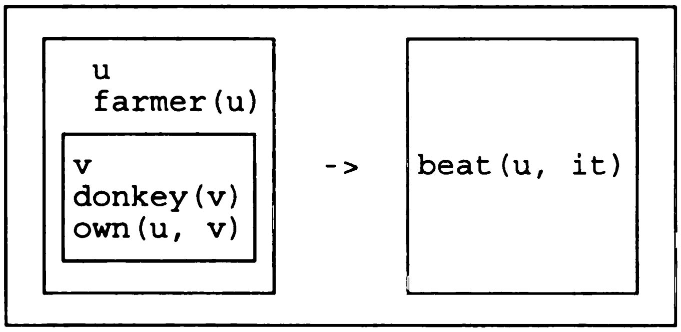
(32) Every person who has a credit card$_i$ will pay his bill with it$_i$.

In (32) the pronoun it can be coreferential with a credit card, and, if so, (32) means that every person will pay his bill with at least and at most one credit card, and does not indicate that every person will pay his bill with his all credit cards. If quantifier every is quantified over a pair of a person and a credit card, (32) will give us a wrong truth condition.

The third problem is that (29) cannot explain the uniqueness presupposition that the pronoun it in (28) carries. That is, (28) should be evaluated with regard to the farmers who have just (and at most) one donkey. Even though a farmer who has 99 donkeys does not beat his own donkeys at all, it does not determine the truth condition of (28). The universal reading of a donkey in (28) cannot express such a uniqueness.

As we have seen so far, in addition to the representation (29), we need another DRS to represent the characteristics described just above. So we can tentatively represent the first and the second characteristics for (28) in the following way:

(33)



Then, how can we maintain the anaphoric relations between E-type pronouns and their antecedents, since <u>a donkey</u> is too deeply embedded in the antecedent DRS?

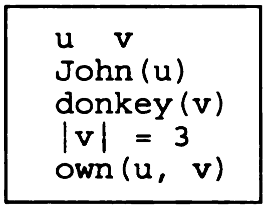
To solve this problem, Kadmon (1990) suggests that the DRS may contain "implicated, accommodated and contextually supplied information". (p. 273) Following Kadmon (1990), a speaker and a hearer can revise the DRS, based upon contextual information. In other words, the DRS's are basically constructed with the syntactic structures as an output of syntactic analysis, and furthermore, to make an utterance felicitous in the context, can be revised with the information which is contextually supplied or specified. Let us take some examples to look at the contextual accommodation, as Kadmon (1990) suggests.

(34) a. John owns three donkeys.

b. John owns three donkeys$_i$. Harry vaccinates them$_i$.

First of all, (34.a) is represented as (35) by a construction rule assumed.

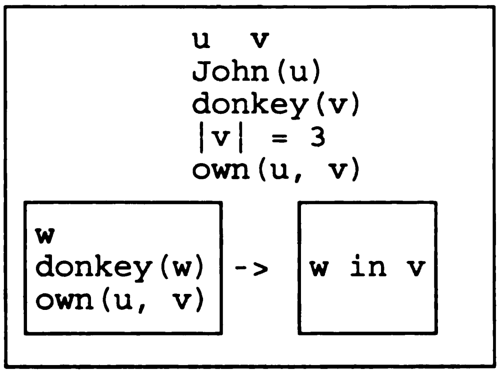
(35)



This DRS expresses that John owns at least three donkeys. That is, among a set of the donkeys that John owns, we may talk about any subset with three donkeys.

However, following Kadmon (1990), (34.a) may be uttered in a context containing a scalar implicature which implies that John does not own more than three donkeys. Then (35) will be revised by a speaker and a hearer as (36).

(36)



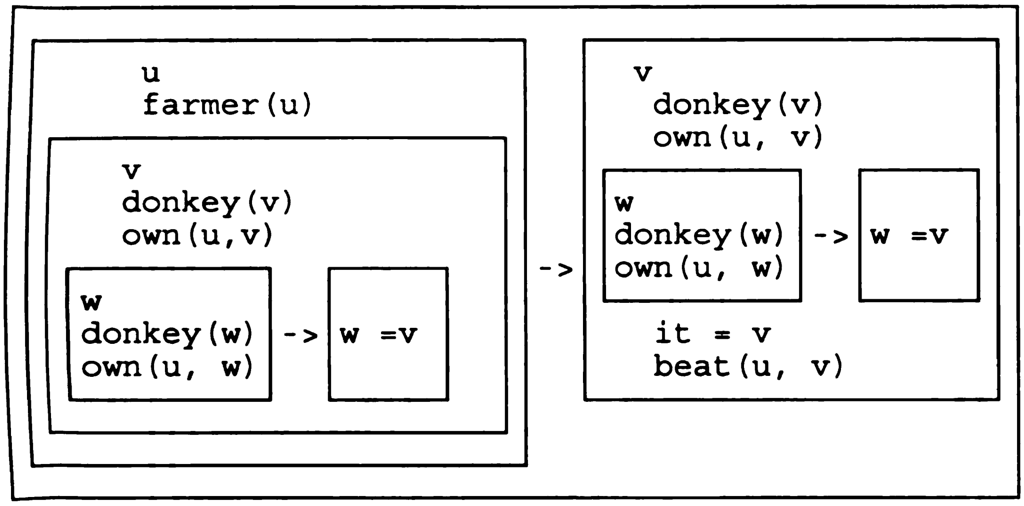
Now let us look at (34.b), a case of E-type pronouns. When a speaker and a hearer processes the first part of (34.b), he constructs a DRS like (35). When he utters or hears the second part, however, (35) cannot be felicitous, if it simply accommodates the second utterance as (37), since (37) does not express the uniquenesss that (34.b) implies. So a speaker and a hearer revise (37) as (38) in processing the second utterance in order to accommodate the uniqueness.

(37)
$$\boxed{\begin{array}{l} u \quad v \\ \text{John}(u) \\ \text{donkey}(v) \\ |v| = 3 \\ \text{own}(u,\ v) \\ \text{Harry}(w) \\ \text{them} = v \\ \text{vaccinates}(w,\ v) \end{array}}$$

(38)
$$\boxed{\begin{array}{l} \qquad u \quad v \quad w \\ \qquad \text{John}(u) \\ \qquad \text{donkey}(v) \\ \qquad |v| = 3 \\ \qquad \text{own}(u,\ v) \\ \qquad \text{Harry}(w) \\ \qquad \text{them} = v \\ \qquad \text{vaccinates}(w,\ v) \\ \boxed{\begin{array}{l} x \\ \text{donkey}(x) \\ \text{own}(u,\ x) \end{array}} \ \to \ \boxed{\ x \text{ in } v\ } \end{array}}$$

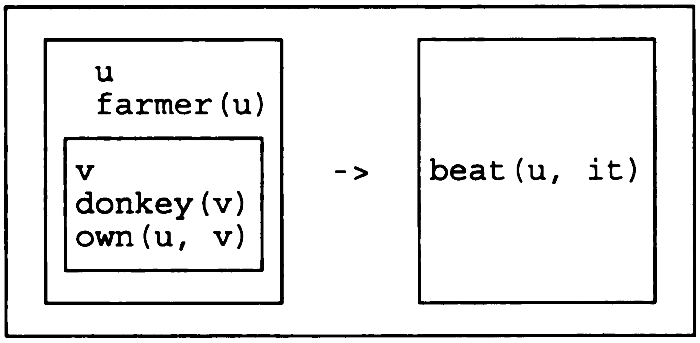
Turning to (28), it should accommodate such a uniqueness in the DRS, and such accommodation also solves the problem in which E-type pronouns are not anaphorically linked to the antecedents. That is, a contextual accommodation can not only make a utterance felicitous but also, by recovering some missing information, provide an antecedent for E-type pronouns which can be anaphorically linked to it. Then (28) can be represented as (39).

(39)

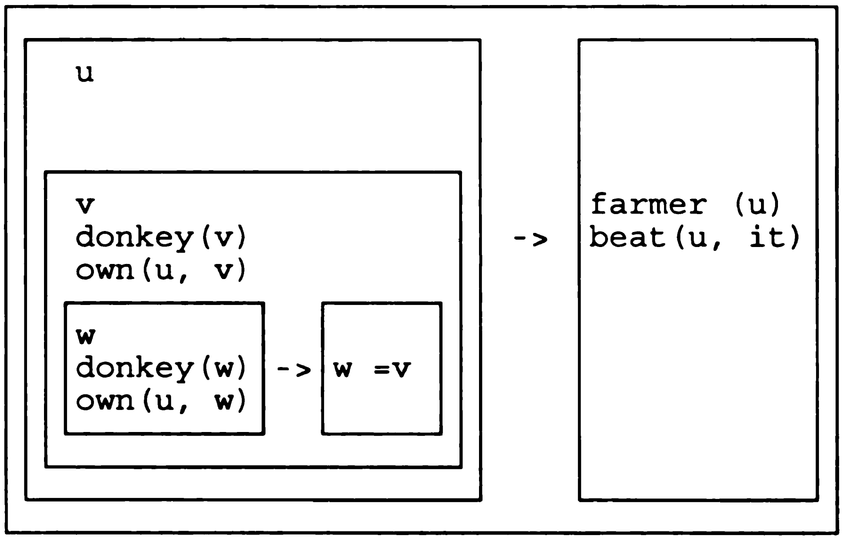


Now let us take a close look at how (39) can be built. First a speaker or a hearer builds (40.a) for (28). A sub-DRS is embedded into the antecedent DRS, since the quantifier <u>every</u> here takes quantification over <u>farmer</u>, not over <u>farmer-donkey</u> pairs. Further, <u>a donkey</u> is unique relative to every farmer. So the accommodation modifies (40.a) as (40.b). Finally, the reconstruction or copy of the sub-DRS in the antecedent DRS into the consequent DRS will build (39) as a result.

(40) a.



b.



Now let us take into consideration the curious copy process mentioned just above. Kadmon (1990) has mentioned that the copy process is required to provide the antecedent for the E-type pronoun in (39), but has not pointed out any constraints on the copy: e.g. when the reconstruction should take place or when it should be blocked. Regarding a constraint on the reconstruction, we may think of uniqueness and/or maximality. That is, the copy should take place iff the antecedent implies uniqueness. In (39) a donkey shows uniqueness and the copy is applicable to it.

With this constraint we can explain such a bad sentence as *Every farmer who owns no donkey$_i$ beats it$_i$. In this sentence no donkey does not imply uniqueness at all, and the copy cannot take place. If the copy is blocked out, the pronoun it cannot have any antecedent, since no donkey is deeply imbedded in the antecedent DRS like in (40.a).

Furthermore, we can also explain such a bad sentence as

*John bought no donkey$_i$, and Harry vaccinated it$_i$. We can represent the sentence as (41).

(41)

```
 ________________________
| u  v                   |
| John(u)                |
|                        |
|      ______________    |
|     |w             |   |
|   ¬ |donkey(w)     |   |
|     |bought(u, w)  |   |
|     |______________|   |
|                        |
| Harry(v)               |
| vaccinated(v, it)      |
|________________________|
```

In (41) the DRS containing the antecedent donkey is subordinate to the DRS containing the pronoun it, and hence donkey cannot be accessible to it. Further, no donkey does not imply uniqueness, and so the copy is also blocked.

In the connection with pragmatic accommodations, a question has been raised whether or not pronominal interpretations are purely based upon such pragmatic or contextual accommodations. Evans (1977) and Heim (1990) argue that some syntactic factors significantly have effect on the interpretation or linking of E-type pronouns and their antecedents. The pragmatic accommodation approach to the e-type pronouns cannot explain the following sentence: (Heim 1990 p.165)

(42) a. Every man who has a wife$_i$ sits next to her$_i$.

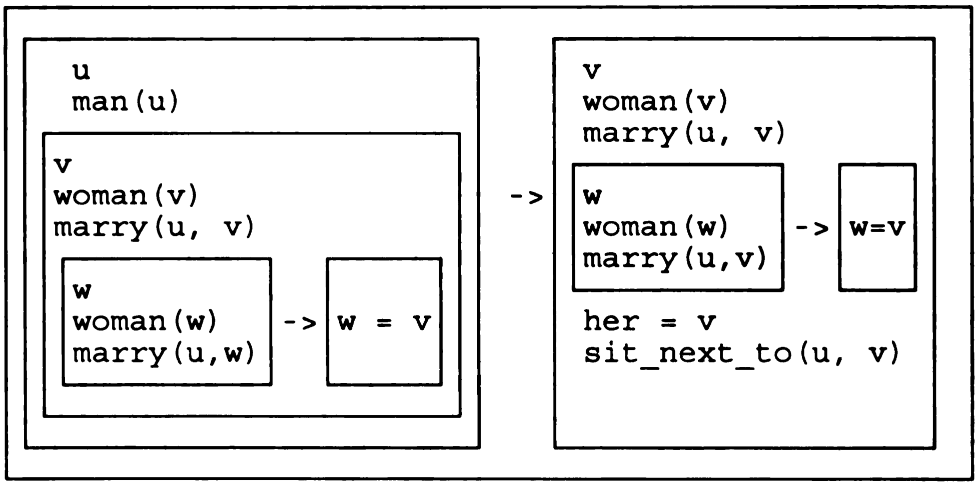
b. *Every married man sits next to her.

(42.a) can be interpreted in a way that a wife is linked to her. Although every married man in (42.b) implies every man who has a wife in (42.a), however, (42.b.) cannot be i

we m
42.I
sente
the (

interpreted like (42.a). If we build the DRS for (42.b), we would get (43)[5]. Here in (43) the pragmatic accommodation would recover some contextual information and provide a candidate for the pronoun. But such interpretation is not natural.

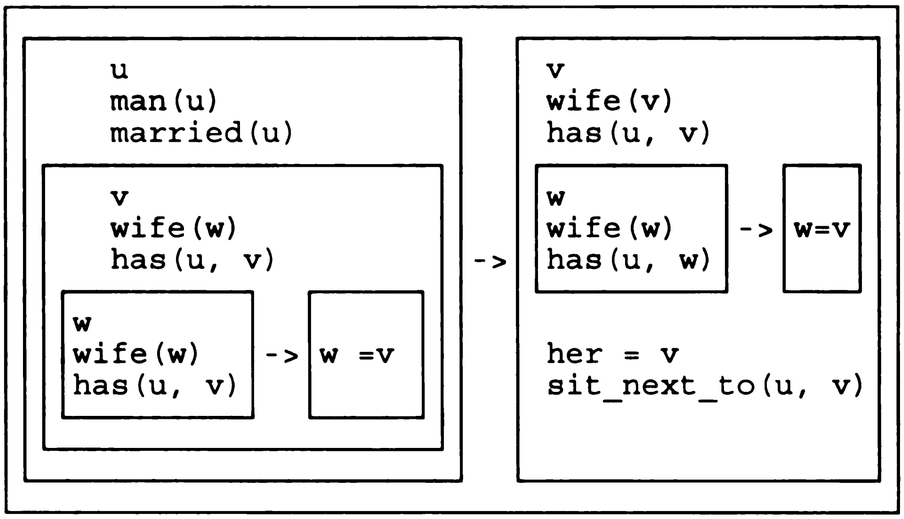
(43) a.*



[5]If we assume that there is an implicit argument for married we may represent (42.b) as (43.b); otherwise we may represent (42.b) as (43.a). Let us assume that (43.b) is correct. Then a sentence like John was killed may be represented as (i) rather than the (ii).

(i) d = <{u}, {John(u), kill(-,u)}>.
(ii) d = <{u} {John(u), killed(u)}>.
A notation '-' in (i) will be explained later in this section.

b.



Now we may take two alternatives into consideration to explain such sentences as (42). The one is to implement a linking mechanism of E-type pronouns and their antecedents in syntax, and the other to put more constraints on pragmatic accommodations.

Evans (1977) and Heim (1990) take the former approach and proposes that E-type pronouns should be syntactically linked with their antecedents and that the well-formedness conditions for E-type pronouns should be formulated in terms of the syntactic antecedent-anaphor relations. Heim (1990) tries to construct E-type pronouns' linking with their antecedents at the syntactic level.

Following Heim (1990), the E-type pronouns are linked with their antecedents by coindexation at LF[6], and this coindexation also constrains the range of the function that

[6]The LF is a syntactic level whish is assumed to be posited after a surface level.

interprets the E-type pronouns. For example, the coindexing in (28) between a donkey and it constrains the function that has a set of donkeys as its range. Heim (1990) treats the E-type pronoun-anaphoric relations with the following transformational rule (P. 170):

(44) X S Y NP_i Z => 1 2 3 4 + 2 5

1 2 3 4 5

conditions: 4 is a pronoun

2 is of the form $[_S\ NP_i\ S]$

6 7

This stipulative rule applies to E-type pronouns at LF and copies the "antecedent (= term 6) and the antecedent's scope (= term 7) into the position of the pronoun (= term 4)". For example, it converts (45.a) into (45.b).

(45) a. $[\text{every}_x$ [farmer(x) that $[[a_y$ donkey(y)$]_2$ [x owns y]]]$]_1$ [x beats it_2].

b. $[\text{every}_x$ [farmer(x) that $[[a_y$ donkey(y)$]_2$ [x owns y]]]$]_1$ [x beats $[it_2$ $[[a_y$ donkey(y)$]_2$ [x owns y]]]].

As already mentioned before, the syntactic level is problematic to represent the pronominal references, since intersentential relations cannot be explained. The transformational rule (44) is problematic. The rule not only specifies the linkage between an antecedent and the pronoun with stipulation, but also has conceptual problem.

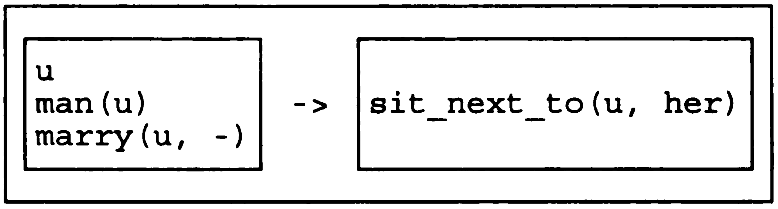
Following (44), the reconstruction or copy process can take place, regardless of how far S (term=2) is from the pronoun (term = 4). It seems to be not true, because pronominal references take place under some kinds of locality conditions. Now there is only one alternative left with us: more constraints on pragmatic accommodation. Let us consider how we can represent (42.b) in DRT.

There might be two options. The one is to prevent the theory from accommodating such contextual information as that in (42.b). That is, a DRS can contain some types of contextual information only--e.g. uniqueness as Kadmon (1990) proposed. The other is to allow the theory to accommodate the contextual information that (42.b) implicitly has, and to make some constraints on the introduction of discourse referents. Leaving for further research the question about which alternative is more adequate in the theory, but here I tentatively take the second alternative.

As I have mentioned before, deictic pronouns, proper nouns, demonstrative NP's and indefinite NP's can introduce discourse referents. Furthermore, if these NP's are explicitly uttered in a linguistic way, they can add new discourse referents to a DRS. More specifically speaking, the discourse referents are introduced not in a contextual or nonlinguistic way but in a purely linguistic way. If a speaker does not linguistically express something about an object, it will never be a discourse referent.

For example, in (42.a) <u>a wife</u> is expressed linguistically and introduces a referent which can be an antecedent for a pronoun her. However, in the case of (42.b), married of every married man cannot introduce a discourse referent to a DRS, because a context or background knowledge cannot introduce any discourse referent. Hence the representations of (43.a) and (43.b) are ill-formed, regardless of the linkage between the pronoun her and v in wife(v) or woman(v). So a speaker or a hearer should construct the DRS (46) for (42.b).

(46) *



The notation '-' of marry(u, -) in (46) means don't care. That is, marry(u, -) means that u is married but we don't care if u marries whoever she is. Truth-conditionally speaking, marry(u, -) is true in a model M iff u belongs to a universe UM and there is a non-empty set of marry relations u has in M.

2.3 Pronouns and Antecedent Search in DRT

Even though Kamp (1981) does not specifically describe how the antecedent is assigned to an anaphor, there is said to be an algorithm to find an antecedent in a set of discourse referent introduced in a DRS in some way.

One of the important factors in determining possible antecedents is the grammatical agreement with respect to person, gender and number.

Another factor is precedence. Reinhart (1983) argues that the linear precedence in a sentence is irrelevant because of the following examples: (p. 34)

(47) Near him$_i$, Dan$_i$ saw a snake.

```
(48)           S
           /   |   \
         PP    NP    VP
         |     |     |
      Near him  Dan  Saw a snake
```

So Reinhart (1983) proposes the c-command rule to interpret the pronominal coreference. She argues that (47) is structurally represented as (48), and the antecedent can c-command its anaphor and be accessible to the anaphor.

However, we cannot explain (49) with c-command rule.

(49) a. His$_i$ wife loves John$_i$.

```
     b.    S
          /     \
         NP       VP
         |      /     \
      his wife  V      NP
                |      |
              loves   John
```

In (49) John cannot c-command his but they can be coreferential in fact. So I will propose that the precedence should be defined in terms of a DRS.

(50) a. An antecedent should precede its anaphor in a DRS.

b. A discourse referent x precedes a discourse referent y iff:

(i) x < y in a universe U in DRS where < is the order of introduction, or

(ii.) a DRS which contains x is superordinate to a DRS which contains y.

(51) A DRS x is superordinate to a DRS y iff

(i) x contains y, or

(ii) x is an antecedent DRS of an If-then DRS whose consequent DRS is y.

(52) Superordination is reflexive, asymmetric, and transitive.

In the case of (47) and (49) the proper nouns are introduced into the top level of that DRS as Kamp (1981) suggests, and then always accessible to the anaphoric pronouns as (53).

(53) a.

```
u  v
Dan(u)
snake(v)
saw_near(n, v, u)
```

b.

```
u  v
John(u)
wife(v)
wife_of(v, u)
love(v,u)
```

Now we might raise a question about the introduction of a discourse referent into the top level of DRT. If it is true, how can we explain the ungrammaticality of (54)? As in (53), Dan places a referent at the top of the DRS, and accessible to he. Then we cannot distinguish (53) from (54) in terms of a DRS at all.

(54) a.* Near Dan_i, He_i saw a snake.

b.
```
u  v
Dan(u)
snake(v)
saw_near(u, v, u)
```

Now let us remind ourselves of the level of DRT. As mentioned in section 1.3 and 2.2, the syntactic analysis is an input to DRT. At the level of syntactic analysis the anaphora like reflexives and reciprocals are determined, and only the disjoint reference, not anaphoric relation, of pronouns are represented. In (54) Dan and he should have disjoint reference; otherwise it would violate the principle C. Then the ungrammaticality of (54) is actually determined at syntactic level, not at DRT level.

Since the concept of precedence is taken in DRT, it is necessary to make further modification of Kamp's DRT. Precedence implies that the discourse referents are ordered in some way; so the universe U should be assumed to be not a simple unordered set of discourse referents but a set of internally ordered referents. This modification can be related to the locality to determine the antecedent-anaphoric relations to be discussed just below.

Another factor to determine the antecedent-anaphoric relations is locality. There might be several types of locality conditions, but I will consider only locality in time. As I described just above, the discourse referents are placed in the order in which they are introduced into the DRS, and the ordering will be kept as the conversation goes on. When more than one antecedent is available, it is assumed that the entity which is referred to the most accessible to the anaphor. Then we can say that the anaphoric function is to reorder the discourse referents already introduced in the discourse by referring to its antecedent.

Let us take an example for this. Let us assume that there is a context in which three referents, u, v, and w, have been introduced which refer to <u>John</u>, <u>Bill</u> and <u>Mary</u>, respectively. Further, they are assumed to have been introduced in that order: <u>Mary</u> is the most recently introduced, and so on. Now a speaker uses a pronoun <u>he</u> to refer to a referent in a context. Then w is the most accessible to the pronoun <u>he</u>, but does not grammatically agree with it. So the next candidate is v now. If there is no other factors to prevent it from being coindexed with the pronoun, the pronoun will refer to that v. Then the reordering of the discourse referents will take place: u, w, and v are in that order. If a speaker uses a pronoun <u>he</u> once more after this, it refers to v again.

Another factor is the common knowledge. Let us take an example for this. When a speaker utters the sentence (55.a), the hearer will understand that Bill was injured if we have the entailment rule (55.b) in the common knowledge or lexicon:

(55) a. John hit Bill, and he was injured.

b. hit(x,y) -> injured(y)

Of course, he may be coreferential with John. However, for a speaker to make a hearer understand (55.a) in that way, the speaker must provide enough information to override the common knowledge expressed in (55.b).

Another factor is a modality of a sentence. Roberts (1989) introduces the modal subordinate constructions with respect to anaphoric relations as follows: (p.683)

(56) a. If John bought a book$_i$ he'll be home reading it$_i$ by now.

b. It$_{j/*i}$ is a mystery story.

(57) a. If John bought a book$_i$ he'll be home reading it$_i$ by now.

b. It$_i$ will be a mystery story.

The sentences in (57) are an example of a modal subordination. The truth value of the sentence (57.b) is determined conditionally in the possible worlds in which (57.a) is true. That is, the modality of (57.a) takes scope over (57.b). For this case we may say that (57.b) is modal-subordinate to (57.a). Since (57.b) is under the modal scope of (57.a), it is possible for the pronoun it in (57.b) to be coindexed with a book in (57.a).

However, the pronoun it in (56.b) cannot be coindexed with a book in (56.b), since (56.b) is a factual sentence without any modal force and cannot be subordinate to (56.a).

3. Computing Discourse Representation Structures

It is not a trivial task to translate surface strings to discourse representation structures (DRS's). In this chapter let us consider some algorithms to construct DRS's from English surface strings as an input.

Computing DRS's requires two separate processes. The first process is to take as an input a discourse (or a set of sentences in a discourse), parse with phrase structure rules and translate the input into DRS's. The second process is to translate DRS's into Prolog facts and rules and embed them in a model to simulate the model-theoretic semantics. In this chapter their implementations will be discussed in terms of Prolog as a logic programming language, based upon Johnson and Klein (1986), Covington and Schmitz (1988), Covington, Nute, Schmitz, and Goodman (1988), and Covington (1994).

3.1 Phrase Structure Rules and Building DRS's

The phrase structure rules are augmented with two types of feature structures: syntactic and semantic feature structures. The syntactic features are used to control the parsing process, and the semantic features to build DRS's during the parsing process. Unification allows the feature structures to be passed from one node to another in a parsing tree. Thus the phrase structure rules should specify the information about word order and the way of unifying feature structures.

To help understanding the parsing procedure, let us describe the phrase structure rules, and then the feature structures and general unification mechanisms, and then, the rules of DRS constructions by unification.

3.1.1 Phrase Structure Rules

The parser here will accept discourses which observe the following phrase structure rules. For simplicity the phrase structure rules here represent only the word order. The phrase structure rules are written in definite clause grammar (DCG) notation. The string in brackets is a terminal string.

```
(1) a.  discourse --> sentence, discourse.
    b.  sentence --> np, vp.
    c.  sentence --> np, aux, vp.
    d.  sentence --> [if], sentence, [then], sentence.
    e.  sentence --> aux, np, vp.
    f.  np --> n.
    g.  np --> [].
    h.  np --> n2, rc.
    i.  np --> n2.
    j.  n2 --> det, n1.
```

```
k.  n1 --> adj, n1.

l.  n1 --> n.

m.  vp --> v, np.

n.  vp --> v.

o.  vp --> adj.

p.  vp --> np.
```

(1.a) helps the parser scan a sentence one by one until the discourse is empty. (1.b-e) are the rules to parse sentences. (1.b-d) represent declarative sentences including copulars. (1.e) is for questions. (1.d) is a rule for if-then sentences.

(1.f-l) are for noun phrases including relative clauses. (1.f) is a rule for proper nouns and pronouns. (1.g) is for a trace or gap in relative clauses. (1.h) is a NP rule including relative clauses. (1.i-l) are for common nouns.

(1.m-p) are the rules for verb phrases. (1.m) is for transitives, and (1.n) is for intransitives. (1.o-p) are for copular predicates.

In the subsequent sections we will describe how the phrase structure rules in (1) accommodate feature structures and their unification.

3.1.2 Feature Structures and Unification

The parser here implements a unification-based grammar, using GULP 3 (Graph Unification Logic Programming) as an extension to Prolog written by Covington (1994). In the unification-based grammar each word in a sentence is assumed to have a set of features. The features specify the function of a word in a sentence. By unification the features are passed through the nodes in the parsing tree. The feature structures and mechanism to unify (or merge) features are thus important.

The parse uses the following partial feature structures to control the parsing process and build DRS's, where a feature structure is represented as a:b and a is an feature attribute and b is the value of that feature: (cf. Covington and Schmitz (1988) p.4)

```
(2) a.  syntactic features[7]
        syn: index: ...
             class: ...
             arg1: ...
             arg2: ...
    b.  semantic features
        sem: in:  ...
             out: ...
             res: in: ...
                  out: ...
             scope: in: ...
```

[7]It is also possible to add other syntactic features such as case, number, person to control the parsing procedures.

out: ...

The roles of the features in (2) are summarized as follows:

index has as its value a discourse referent instantiated on nouns (common and proper nouns).

class has as its value common/proper, instantiated on nouns, or transitive/intransitive, instantiated on verbs.

arg1 and arg2 are the discourse referents of subject and direct object as arguments of a predicate, being unified with their syn:index features.

sem:in is a list of one or more DRS's which are currently being processed.

sem:out is the DRS after processing the current node.

sem:res and sem:scope are used for the logical structure of the sentence.

In general, following Covington (1994), the unification-based grammar uses the following mechanism of the unification of feature structures A and B giving C:

(i) If a feature x occurs either in A or B exclusively, then it also occurs in C with the same value.

(ii) If a feature x occurs in both A and B, then it also occurs in C, and its value in C is the unifier of its values in A and B.

Now feature values are unified as follows:

(i) If both A and B have atomic symbols as their values, then their values must be the same atomic symbol.

(ii) If A has a variable as its value and B has a variable as its value, then B has the same value as A.

(iii) If both A and B have variables as their values, then the variables become the same variable.

(iv) If both A and B have feature structures as their values, they unify by applying this process recursively.

To pass feature structures from one node to another in a parsing tree, a feature is splitted into two sub-features, one for input and the other for output. Consider the rule for example: (Covington (1994) p.12)

```
(3)  S              ->      NP                    VP
 [sem: [in:X1] ]   [sem: [in:X1] ] [sem: [in:X2] ]
 [     [out:X3]]   [     [out:X2]] [     [out:X3]]
```

This rule specifies that the sem of the S has some initial value of X1. X1 is then passed to the NP, which modifies it in some way and gives X2 out. Then X2 is passed to the VP for further modification. The output of the VP is X3, and it becomes the output of the S. This feature passing is used to build DRS's in parsing.

3.1.3 Representing DRS's

A DRS is a set of discourse referents U and a set of conditions C. It can be implemented by the following Prolog predicate:

(4) drs(U,C).

U is a list of discourse referents and C is a list of predicates of these referents. Discourse referents are represented with unique Prolog variables at the level of DRT. When DRS's are embedded into a model, they will be, if necessary, skolemized by instantiating uninstantiated Prolog variables with unique integer number.

The DRS for questions and if-then sentences are represented as query(drs(U,C)) and if_then(drs(U1,C1), drs(U2,C2)) where drs(U1,C1) is the antecedent and drs(U2,C2) the consequent, although the DRS for declarative sentences can be simply represented by (4).

In the next section let us consider how the phrase structure rules accommodate feature structures and their unification, and how the parser builds DRS's during the parsing process.

3.1.4 Parsing Discourses

A discourse is a set of sequent sentences. As a sentence in a discourse is parsed, the parsing rules pass the discourse information from one rule to the next rule using unification. We can rewrite (1.a), incorporating feature structures, as in (5).

```
(5) parse_discourse(D) -->
         { D = sem:in:X,
           S = sem:in:X,
                  S = sem:out:Y,
                  D1 = sem:in:Y,
                  D1 = sem:out:Z,
                  D = sem:out:Z
                },
                parse_sentence(S,[],[]),
                endpunct, {!},
                parse_discourse(D1).
          parse_discourse(D) -->
                { D = sem:in:X,
                  D = sem:out:X }.
          endpunct --> ['.'] ; ['?'].
```

The rule (5) will do the following things:

(i) If the discourse has one or more sentences, then parse_discourse passes all the semantic features of sem:in to parse_sentence.

(ii) Next, the rule parse_sentence (to be described later) modifies sem:in, and passes it into parse_discourse for further modification.

(iii) If the discourse has no sentence, then its current semantics will be the output of parse_discourse.

While this procedure is recursively going on, sem:out of parse_discourse accumulatively collects all information the parsing rules have modified. Initially, the semantic feature sem:in will contain an empty DRS in the form of drs([],[]).

The phrase structure rule (1.b) for a declarative sentence can be written as follows.

```
(6) parse_sentence(S,H1,H3) -->
            { S = sem:X,
              NP = sem:X,
              NP = sem:scope:Y,
              VP = sem:Y,
              NP = syn:index:Z,
              VP = syn:arg1:Z
            },
            parse_np(NP,H1,H2),
            parse_vp(VP,H2,H3).
```

This rule will do the following things:

(i) The rule parse_sentence passes all the incoming semantic information to the NP rule, parse_np, and at the same time the output semantics of the NP is the semantics of the sentence rule.

(ii) The VP is the scope of the NP.[8]

(iii) The index (or the discourse referent) of the NP is the same as the subject or arg1 of the verb. So arg1 has the same value of the index of this NP.

We can write a parsing rule (1.e) for a question as follows:

```
(7) parse_sentence(S,H1,H3) -->
```

[8]The scope of the NP is, in fact, that of the determiner. It will be clear when we describe the semantics of determiners.

```
            { S = sem:in:X,
              X = [drs([],[])],
              NP = sem:in:X,
              NP = sem:scope:Y,
              VP = sem:Y,
              NP = syn:index:Z,
              VP = syn:arg1:Z,
              NP = sem:out:[DRS],
              S = sem:out:[drs([],[query(DRS)])]
            },
            parse_aux(_),
            parse_np(NP,H1,H2),
            parse_vp(VP,H2,H3).
```

(7) will do the following things in addition to (6):

(i) It checks whether the incoming DRS is empty. That is, it checks if a question is embedded in a discourse. If it were, it would fail. Otherwise, this topmost DRS is added, and passed to the NP rule. The NP and VP rule will modify this empty DRS.

(ii) The DRS is then embedded into the original DRS, forming query(drs) to represent a question.

This technique can generally apply to building sub_DRS's including if-then sentences. The following phrase structure rule is for if-then sentences.

```
(8) parse_sentence(S,[],[]) -->
            { S = sem:in:X,
          S1 = sem:in:[drs([],[])|X],
          S1 = sem:out:Y,
          S2 = sem:in:[drs([],[])|Y],
          S2 = sem:out:[DRS2,DRS1,drs(U,C)|Sup_drs],
          S = sem:out:
              [drs(U,[if_then(DRS1,DRS2)|C])|Sup_drs]
        },
        [if],
        parse_sentence(S1,[],[]),
        [then],
        parse_sentence(S2,[],[]).
```

This rule basically works in the same way as (7). It inserts an empty DRS and then constructs the antecedent by passing it to parse_sentence. After that it inserts another empty DRS and builds the consequent in it, passing it to parse_sentence. IF both the antecedent and the consequent DRS are constructed, then this rule combines them into a form of if_then(drs1,drs2), and embeds it in the condition of the original DRS.

We can implement other sentences in (1) with feature structures in the same manner as described above.

3.1.5 Parsing Noun Phrases

The following phrase structure rule is to parse the common noun phrases as in (1.j).

```
(9) parse_n2(N2,H1,H2) -->
          { N1 = syn:class:common,
            N2 = syn:X,
            N1 = syn:X,
            Det = sem:Y,
            N2 = sem:Y,
            N1 = sem:Z,
            Det = sem:res:Z
          },
          parse_det(Det),
          parse_n1(N1,H1,H2).
```

This rule will do the following things:

(i) The syntactic class is specified as common.

(ii) The semantics of the NP becomes the restrictor of the determiner.

To make (ii) more clear, let us consider the semantics of determiners. Following Johnson and Klein (1986), semantically, a determiner has two arguments: restrictor and scope. For example, the sentence A donkey is old. can be translated into predicate logic as (a X: donkey(X)) old(X) where a X is the quantifier, donkey(X) is the restrictor, and old(X) is the scope.

Accordingly, the DRS for A donkey is old. is constructed by passing to the determiner a the NP donkey for the restrictor and the VP old for the scope.

The phrase structure rule of the determiner a is written as follows:

```
     (10) parse_det(Det) -->
               ([a] ; [an]),
               { Det = sem:in:X,
                 Det = sem:res:in:X,
                 Det = sem:res:out:Y,
                 Det = sem:scope:in:Y,
                 Det = sem:scope:out:Z,
                 Det = sem:out:Z
               }.
```

That is, this rule passes the semantics of a determiner to the restrictor, then to the scope. The determiner a has no special representation at all.

However, a determiner like every constructs a DRS in a more complex way:

```
     (11) parse_det(Det) -->
                [every],
                { Det = sem:in:X,
                  Det = sem:res:in:[drs([],[])|X],
                  Det = sem:res:out:Y,
                  Det = sem:scope:in:[drs([],[])|Y],
                  Det = sem:scope:out:
                        [Scope,Res,drs(U,C)|Sup_drs],
                  Det = sem:out:
                        [drs(U,
                        [if_then(Res,Scope)|C])|Sup_drs]
```

```
            }.
```

This rule works the same way as (8) for if-then sentences.

Next, a lexical insertion rule for common nouns can be represented in (12).

```
(12) parse_noun(N) -->
            [Word],
            { common_noun(Word,I,Sem),
              append(Sem,C,NewC),
              N = syn:index:I,
              N = syn:class:common,
              N = sem:in:[drs(U,C)|Sup_drs],
              N = sem:out:[drs([I|U],NewC)|Sup_drs]
            }.
```

This rule will do the following things:

(i) It generates a unique variable I for the common noun.

(ii) It adds I to the list U, and adds the semantics of the common noun to the list C.

It applies to the entire category of common nouns. The semantics for a word man is represented in (13).

```
(13) common_noun(man,X,[man(X),gender(X,m)] ).
```

This rule forms two predicates man and gender. The latter is used for antecedent search.

3.1.6 Parsing Verb Phrases

The phrase structure rule (1.m) for the transitive VP can be represented as follows:

```
(14) parse_vp(VP,H1,H2) -->
          { V = syn:class:transitive,
           VP = syn:X,
           V = syn:X,
           NP = sem:Y,
           VP = sem:Y,
           NP = syn:index:Z,
           VP = syn:arg2:Z,
           V = sem:W,
           NP = sem:scope:W
          },
          parse_verb(V),
          parse_np(NP,H1,H2).
```

This rule will do the following things:

(i) It specifies that this VP is a transitive.

(ii) The index of this NP becomes the value of the arg2 or direct object of this verb.

(iii) The semantics of this VP is the scope of this NP.

The lexical insertion rule for transitive verbs can be implemented as in follows:

```
(15) parse_verb(V) -->
          [Word],
          { transitive_verb(Word,A1,A2,Sem),
          append(Sem,C,NewC),
        V = syn:class:transitive,
        V = syn:arg1:A1,
        V = syn:arg2:A2,
        V = sem:in:[drs(U,C)|Sup_drs],
        V = sem:out:[drs(U,NewC)|Sup_drs]
    }.
```

The lexical item for a transitive verb <u>own</u> is represented as in the following:

(16) transitive_verb(owns, X,Y,[own(X,Y)]).

This rule forms the predicate, using the indices that are passed to it as syntactic arguments, and insert the predicate in the condition of the current DRS. Thus the verb rules depend on other rules to pass the values of arg1 and arg2 to them.

3.2 Embedding DRS's in Knowledge Base

To evaluate the truth of a DRS, we should embed it into a model or knowledge base. It requires two processes. One is to clean up some information from the DRS, which is temporary at the level of DRT for some reason, and the other to translate the DRS to appropriate Prolog facts and rules. This section will consider the latter process only: how a DRS can be translated into Prolog facts and rules, and embedded into a model, based upon Covington, Nute, Schmitz, and Goodman (1988).

3.2.1 Simple Statements

The statements about facts such as declarative sentences should be added to the knowledge base for evaluating/querying them later. However, it has a problem if we add their DRS's directly to the knowledge base.

For example, a sentence <u>A farmer owns a donkey.</u> can be translated into the following DRS:

(17) DRS([X,Y], [own(X,Y), farmer(X), donkey(Y)]).

We cannot add the DRS (17) itself to the knowledge base, leaving the variables X and Y free. For Prolog facts the free variables are implicitly interpreted as universal quantifiers. So (17) would be interpreted in such a way that any farmer owns any donkey.

To solve this problem, a unique value must be assigned to an existentially quantified variable. To do this we can assign a unique integer to it. That is, we can assign the free variables x and y [1] and [2], respectively, and embed the facts:

(18) farmer([1]). donkey([2]). owns([1], [2]).

This is a special process of skolemization in which existentially quantified variables are replaced with unique identifiers.

3.2.2 If-then Sentences

Fo

is

r

f

c

A simple if-then DRS is equivalent to a Prolog rule. For example the DRS for the sentence Every donkey is old. is represented as:

```
(19) if_then(drs([X],[donkey(X)]), drs([],[old(X)])).
```

This is equivalent to the Prolog rule:

```
(20) old(X) :- donkey(X).
```

We may have a problem if the consequent, e.g. the rightside DRS contains more than one predicate, as in the following:

(21) Every donkey is old and gray.

```
(22) if_then(drs([X],[donkey(X)]),
               drs([],[old(X),gray(X)])).
```

A Prolog rule cannot have two or more predicates in its consequent; hence (23) is not permitted in Prolog.

```
(23) old(X), gray(X) :- donkey(X).
```

When this if-then is embedded, the consequent should be distributed into a series of ordinary Prolog rules. For example, a rule of the form a, b, c, :- d, e, f is added to the knowledge base as the three rules:

```
a :- d, e, f.

b :- d, e, f.

c :- d, e, f.
```

If the consequent of an if-then DRS introduces a new variable, it is interpreted as an implicit existential quantifier in Prolog. This is another representational problem in Prolog. For example the DRS for the sentence

Every farmer owns a donkey. is represented as follows:

```
(24) if_then(drs([X],[farmer(X)]),
             drs([Y],[own(X,Y),donkey(Y)])).
```

We might translate (24) to (25).

(25) donkey(Y), own(X,Y) ::- farmer(X).[9]

However, (25) means that every farmer owns every donkey.

If we assign the donkey a unique integer as in (26), then it means that every farmer owns the same donkey identified as [2].

(26) donkey([2]), own(X,[2]) ::- farmer(X).

To represent the meaning that every farmer owns a different donkey, it should be represented as follows:

(27) donkey([2|X]), own(X,[2|X]) ::- farmer(X).

The donkey has a dummy name that contains X so that the dummy name depends on the assignment of X. Then if we assign a farmer X with an integer [1], we can get a donkey named [2,1]; if we assign another farmer X with [3], we can then get a donkey named [2,3]; and so on. This is a process of skolemization.

3.2.3 Questions

It is easy to translate query(drs) for questions into

[9]The symbol ::- is used temporarily to represent the complex if-then DRS's at the level of DRT.

Prolog. For example, the sentence <u>Does a farmer own a donkey?</u> is translated into a DRS:

```
(28) query(drs([X,Y], [own(X,Y), farmer(X),
     donkey(Y)])).
```

Questions are understood as to evaluate the truth of a DRS in the model defined by the current knowledge base. So, we want to know whether there is an assignment of values to X and Y such that farmer(X), donkey(Y), and own(X,Y) will all be true. For Prolog queries the free variables have the meaning of implicit existential quantifiers. So it does not require any skolemization at all. Thus (28) can be translated into a Prolog query as follows:

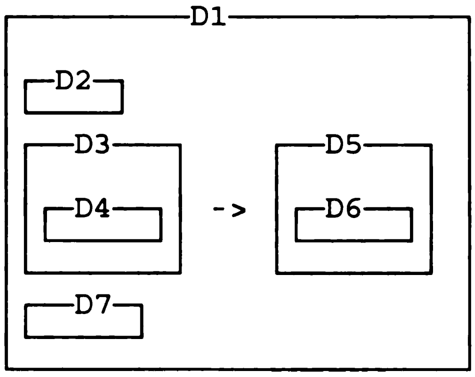
```
(29) farmer(X), donkey(y), own(X,Y).
```

3.3 Anaphoric Pronouns

3.3.1 Antecedent Search

As we have discussed in chapter 2, DRT searches in U of drs(U,C) for an antecedent that agrees with the anaphor in person, gender and number. DRT begins to search for antecedents in the current DRS, and then goes up through superordinate DRS's of the current DRS.

Consider the following hypothetical DRS:

(30)


In (30) D1 is superordinate to all DRS's D2-D7 which it contains. D3 is superordinate to D4, D5 and D6. D5 is superordinate to D6. D2, D4, D6 and D7 are not superordinate to other DRS's.

If a pronoun is in D1, DRT will search for its antecedent in D1. If it is in D5, DRT will search in D5, D3 and D1. If it is in D6, DRT will search in D6, D5, D3 and D1. If it is in D2 DRT will search in D2 and D1. If it is in D4 DRT will search in D4, D3 and D1. If it is in D7 DRT will search in D7 and D1.

In this implementation the DRS construction algorithm works with a list of DRS's. This list begins with the current DRS under construction, followed by all superordinate DRS's, so that all discourse referents in the superordinate DRS's are accessible to the current DRS.

Consider (30) for instance. At the moment D6 is being constructed. The active DRS is D6, and three other DRS's, D1, D3, and D5, are superordinate to it. Thus the DRS list which the construction algorithm can work with at that moment will be as follows:

(31) [D6, D5, D3([...],[D4]), D1([...],[D2,...])].

Note that D2 and D4 are already constructed, closed and added into the conditions of D1 and D3, respectively.

Under the DRS constructions The sem:in feature is a list containing the current DRS plus all DRS's that are superordinate to it rather than a single DRS. So the antecedent of an anaphoric pronoun is found by searching the list of discourse referents in each DRS in the DRS list. This is done by the following rule for pronouns:

```
(32) parse_np(NP,H,H) -->
          [he],
          { NP = sem:in:DRSlist,
            member(drs (U, C), DRSlist),
            member(Index,U),
            member(gender(Index1,m),C),
            Index==Index1,
            NP = syn:index:Index,
            NP = sem:scope:in:DRSlist,
            NP = sem:scope:out:Result,
            NP = sem:out:Result
          }.
```

The rule will do the following things:

(i) It selects drs(U,C) from the DRS list in the order from the nearest superordinate DRS to the farthest.

(ii) It selects a discourse referent Index in U in the order from the most recently introduced entity to the least.

(iii) It checks the gender. If it matches, the rule will set the index of the current NP equal to the index of the antecedent just found.

3.3.2 Subordinations and Anaphora

The basic mechanism described in the previous section can in many cases find an antecedent for pronouns (1.i-iii) described in chapter 2. However, this antecedent search mechanism has some problem with an anaphor which is under a modal/quantificational subdorinations. For example, consider (33).

(33) Every farmer owns a donkey$_i$. It$_i$ is old.

To construct the DRS for (33), the DRS building algorithm takes the following steps:

(i) An empty DRS drs([],[]) is passed to parse_discourse. So the list contains

```
[
 drs([], [])
].
```

(ii) The antecedent DRS, drs([X],[farmer(X)]) is constructed, and the list contains

```
[
 drs([], [])
 drs([X], [farmer(X)]),
].
```

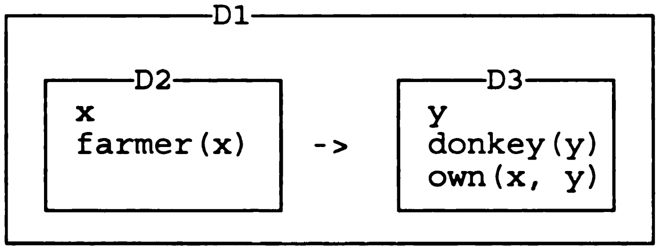
(iii) The consequent DRS, drs([Y], [donkey(Y), own(X,Y)]) is constructed, and the list contains:

```
[
 drs([], []),
 drs([X], [farmer(X)]),
 drs([Y], [donkey(Y), own(X,Y)]),
].
```

(iv) The antecedent and consequent DRS's are combined into the if-then DRS if_then(drs([X], [farmer(X)]), drs([Y], [donkey(Y), own(X,Y)])), and embedded into the topmost DRS, and the list contains:

```
[
 drs([], [if_then(drs([X], [farmer(X)]), drs([Y],
 [donkey(Y), own(X,Y)]))])
].
```

The currently constructed DRS can be diagrammed in (34).

(34) 

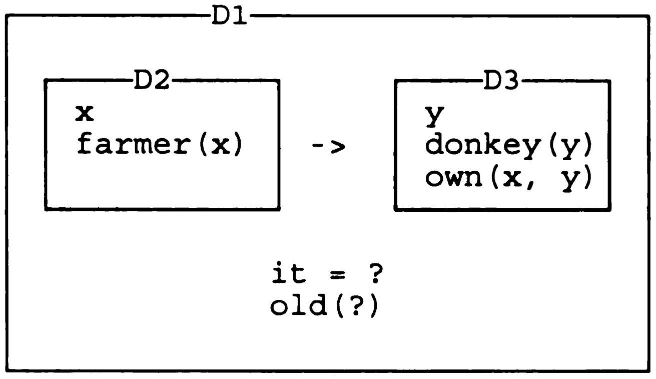
Next, DRT parses the sentence <u>it is old.</u> in the discourse. In this case DRT meets the pronoun <u>it</u>, and tries to find its antecedent in the DRS list. However, the algorithm fails to find an antecedent for the pronoun <u>it</u>, since the DRS list contains only the topmost DRS. Thus the

DRS

135

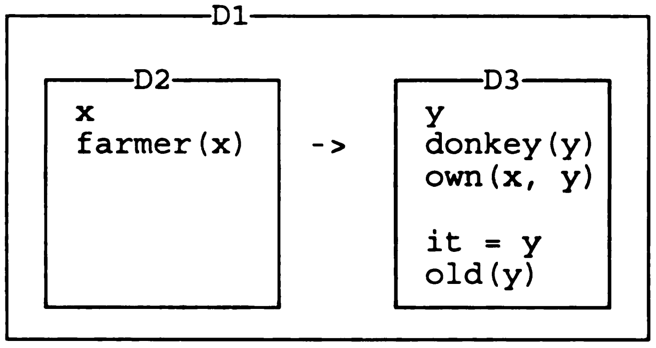
DRS construction algorithm wrongly builds the complete DRS (35) for (33).

(35)



To represent (33) correctly, we expect (36).

(36)



This problem is due to the DRS building procedure rather than to the search procedure. After the algorithm processes a sentence it will close all other DRS's but the topmost DRS. DRT should be able to reopen the closed DRS for antecedent search. This is done by the following rules:

(37)

```
parse_sentence(S,[],[]) -->

    {S = sem:in:DRS,

     DRS = [drs(U,[if_then(DRS1,DRS2)|C])|Sup_drs],

     S1 = sem:in:[DRS2,DRS1,drs(U,C)|Sup_drs],

     S1 = sem:out:[DRS3|_],
```

co

st

ar

th

t

m

p

m

c

```
 S = sem:out:
      [drs(U,[if_then(DRS1,DRS3)|C])|Sup_drs]
},
parse_sentence(S1,[],[]).
```

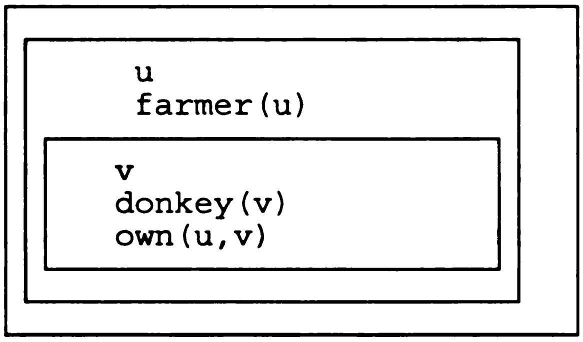
This rule opens the already closed if-then and its consequent DRS, makes the consequent DRS a current DRS, and starts searching for an antecedent. If DRT finds the antecedent in this DRS, then it will insert the analysis of this sentence in it.

3.3.3 The Accommodations and Problems

To represent its DRS for the existential reading of E-type pronouns, we should implement the accommodation mechanism, as described in chapter 2. However, we have some problems with the implementation of the accommodation mechanism in DRT with two respects: The one is that it is difficult to know which predicates should be accommodated for the E-type pronouns. The other is to translate into Prolog the complex if-then DRS's which the accommodation process builds.

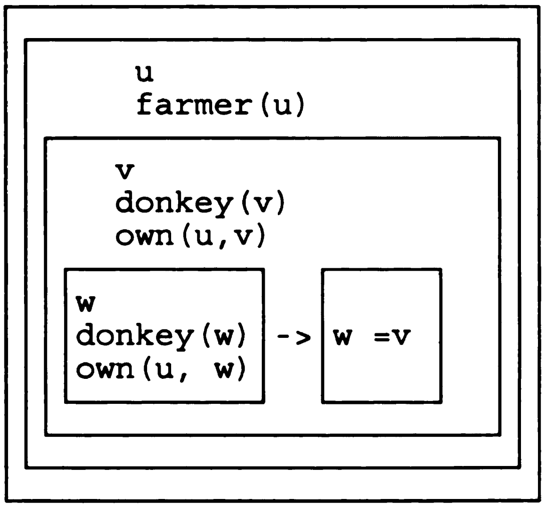
Regarding the former problem, let us consider the DRS for a sentence Every farmer who owns a donkey beats it. The algorithm first constructs the antecedent as follows:

(38)



After that the consequent DRS is constructed. At this point the algorithm fails to search the antecedent of the pronoun it, since the potential antecedent is deeply embedded in the antecedent DRS of the if-then DRS. For the next option DRT reopens the DRS immediately embedded in the just previous superordinate DRS, and search for a potential antecedent. Now the algorithm can find the antecedent donkey for it. However, we should update the DRS containing donkey by accommodating the following DRS for the correct representation and interpretation.

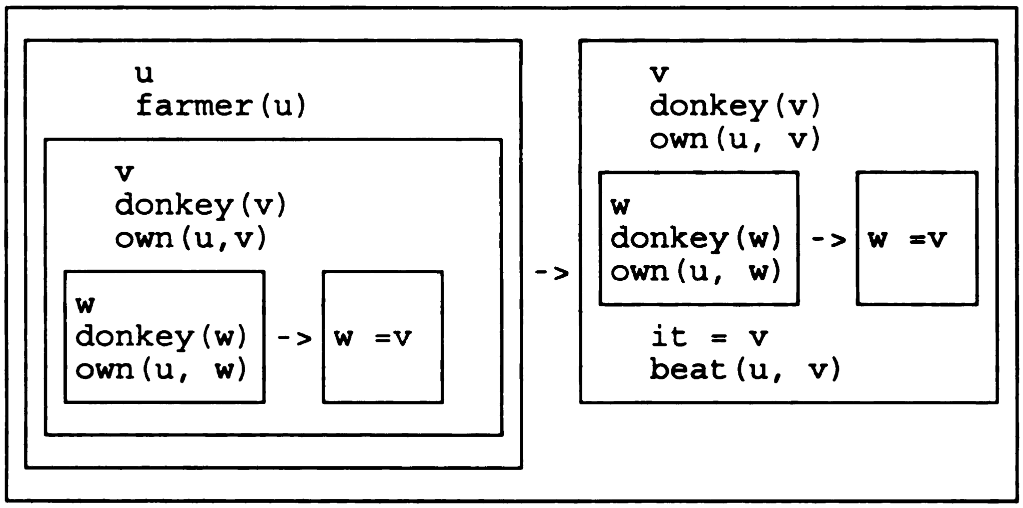
(39)



To accommodate (39), we should know what are the predicates to be accommodated: in this case they are donkey(V) and own(U,V). However, it is difficult to know the name of the predicates in Prolog.

If we could build a DRS for the above example, as in (40), we would meet with the latter problem.

(40)



The DRS (40) has a complex if-then: the antecedent contains another if-then DRS, and the consequent also contains another if-then DRS. Such a complex if-then DRS is hard to be translated into Prolog rules.

4. Conclusion and Further Research

In this thesis I have discussed the pronominal classification, representation and interpretation in DRT, and their implementations in Prolog.

First of all, I have discussed Kamp's (1981) DRT, assuming that DRT is a linguistic level as an interface between the syntactic analysis and model-theoretic semantics. Although syntax analyzes some types of scope and anaphoric relations like reflexive and reciprocal anaphora within a sentence, it is not adequate to represent intersentential scope and anaphoric relations such as modal subordinations, quantificational subordinations, and pronominal references. On the other hand, DRT is adequate in building intersentential relations in a discourse, providing a unified discourse structure.

To understand pronominal phenomena we have reviewed pronominal uses, and classified the pronouns into deictic and anaphoric pronouns in terms of discourse referents at DRT. Deictic pronouns introduce discourse referents, and anaphoric pronouns select as an antecedent one of the discourse referents already introduced in a DRS.

The well-formedness of some types of anaphora should be determined at syntax. If an anaphor is reflexive or reciprocal, the well-formedness is determined by explicit constraints on how they should be coindexed, and if an anaphor is pronominal the well-formedness is determined by expressing how they should not be coindexed.

To determine the antecedents of anaphoric pronouns, we should specify some factors such as grammatical agreements, a hierarchical relations between discourse referents, a local relations between an anaphor and its antecedent, pragmatic accommodations, inferences with common knowledge, and so on.

Of anaphoric pronouns, specially, the properties of E-type pronouns are very complicated. To understand the problems of E-type pronouns clearly, we have reviewed other researchers' works in comparison, and discussed the ways in which the logical forms of E-type pronouns are derived and interpreted. For E-type pronouns the syntactic analysis of E-type pronouns is more problematic, and so DRT should be able to resolve E-type pronouns with the accommodation of pragmatic or contextual information.

To demonstrate the mechanism of the pronominal references in DRT, first of all, we have discussed some algorithms and techniques to translate English surface strings into a DRS in Prolog, and to embed them into a model for evaluation. Although this implementation can resolve pronominal references in many cases, it still has some problems with E-type pronouns in two respects: a predicate identification and a complex if-then embedding. Relating to the former problems, one possible solution is to develop another knowledge representation in Prolog. For example, we may represent the sentence A farmer owns a donkey in Prolog as follows:

```
(1) pred('farmer',[1]),
    pred('donkey',[2]),
    pred2('own',[1],[2])
```

rather than:

```
(2) farmer([1]), donkey([2]), own([1],[2]).
```

We will leave this development open for further research.

We have discussed how to represent pronouns and find their potential antecedents in DRT. But we did not mention anything about which one of the potential antecedents is a real antecedent for a pronoun. Following Sidner (1983) and Grosz (1986), we may be able to find out a real antecedent for a pronoun if we can determine a focus in a discourse. The idea is that a pronoun typically refers to a discourse focus. If it is correct, we can efficiently find a real antecedent for a pronoun if we develop some algorithm to find a discourse focus.

Many factors seem to be involved in determining a discourse focus. Syntax, semantics, inferential knowledge, pragmatic knowledge, and some additional phonological stress are required to detect a current discourse focus and focus shift. Conversely, pronouns sometimes play an important role in guessing a discourse focus. If DRT can somehow accommodate some of these factors it will have a great effect on pronominal references. It will be left for further research.

APPENDIX

This appendix presents some codes to build DRS's and search for pronominal antecedents.

```
/* phrase structure and DRS construction rules */

/* Discourse --> Sentence, Discourse */
parse_discourse(D) -->
     { D = sem:in:X,
       S = sem:in:X,
       S = sem:out:Y,
       D1 = sem:in:Y,
       D1 = sem:out:Z,
       D = sem:out:Z
     },
     parse_sentence(S,[],[]),
     endpunct, {!},
     parse_discourse(D1).

parse_discourse(D) -->
```

```
     { D = sem:in:X,
       D = sem:out:X }.

endpunct --> ['.'] ; ['?'].




/* rules for sentences */
/* We should have implemented the Subj-Verb agreement
* (number and person), but it is not our purpose here. */

/* S --> NP, VP */
parse_sentence(S,H1,H3) -->
     { S = sem:X,
       NP = sem:X,
       NP = sem:scope:Y,
       VP = sem:Y,
       NP = syn:index:Z,
       VP = syn:arg1:Z
     },
     parse_np(NP,H1,H2),
     parse_vp(VP,H2,H3).

/* S --> NP, Aux, VP */
parse_sentence(S,H1,H3) -->
     { S = sem:X,
          NP = sem:X,
          NP = sem:scope:Y,
          VP = sem:Y,
          NP = syn:index:Z,
          VP = syn:arg1:Z
        },
        parse_np(NP,H1,H2),
        parse_aux(_),
        parse_vp(VP,H2,H3).

/* S --> [if], S, [then], S */
parse_sentence(S,[],[]) -->
        { S = sem:in:X,
          S1 = sem:in:[drs([],[])|X],
          S1 = sem:out:Y,
          S2 = sem:in:[drs([],[])|Y],
          S2 = sem:out:[DRS2,DRS1,drs(U,C)|Sup_drs],
          S = sem:out:[drs(U,[if_then(DRS1,DRS2)|C])|Sup_drs]
        },
        [if],
        parse_sentence(S1,[],[]),
        [then],
        parse_sentence(S2,[],[]).

/* S --> Aux, NP, VP */
parse_sentence(S,H1,H3) -->
```

```
    { S = sem:in:X,
      X = [drs([],[])],
      NP = sem:in:X,
      NP = sem:scope:Y,
      VP = sem:Y,
      NP = syn:index:Z,
      VP = syn:arg1:Z,
      NP = sem:out:[DRS],
      S = sem:out:[drs([],[query(DRS)])]
    },
    parse_aux(_),
    parse_np(NP,H1,H2),
    parse_vp(VP,H2,H3).

/* pronominal antecedent search in subordinations */
parse_sentence(S,[],[]) -->
    { S = sem:in:DRS,
      DRS = [drs(U,[if_then(DRS1,DRS2)|C])|Sup_drs],
      S1 = sem:in:[DRS2,DRS1,drs(U,C)|Sup_drs],
      S1 = sem:out:[DRS3|_],
      S = sem:out:[drs(U,[if_then(DRS1,DRS3)|C])|Sup_drs]
    },
    parse_sentence(S1,[],[]).

/* rules for noun phrases */
/* NP --> N */
parse_np(NP,H,H) -->
     { N = syn:class:proper,
       NP = syn:X,
       N = syn:X,
       NP = sem:res:Y,
       N = sem:Y,
       NP = sem:in:Z,
       NP = sem:res:in:Z,
       NP = sem:res:out:W,
       NP = sem:scope:in:W,
       NP = sem:scope:out:Z1,
       NP = sem:out:Z1
     },
     parse_noun(N).

/* pronouns */
/* This routine will search for pronominal antecedents
*which agree in gender.  If we implement person and number
* features for nouns, then we modify this, too. */
parse_np(NP,H,H) -->
     ([he];[him]),
     { NP=sem:in:DrsList,
       member(drs(U,C),DrsList),
       member(Index,U),
       member(gender(Index1,m),C),
          Index == Index1,
          NP = syn:index:Index,
          NP = sem:scope:in:DrsList,
          NP = sem:scope:out:Drs,
          NP = sem:out:Drs
        }.

parse_np(NP,H,H) -->
        [it],
        { NP=sem:in:DrsList,
          member(drs(U,C),DrsList),
          member(Index,U),
          member(gender(Index1,n),C),
          Index == Index1,
          NP = syn:index:Index,
          NP = sem:scope:in:DrsList,
          NP = sem:scope:out:Drs,
          NP = sem:out:Drs
        }.
/* We can add more pronouns hereafter. */

/* a trace in relative clauses */
/* NP --> [] */
parse_np(NP,[rel(Index)|Rest],Rest) -->
        [],
        { NP = sem:in:X,
          NP = sem:scope:in:X,
          NP = sem:scope:out:Y,
          NP = sem:out:Y,
          NP = syn:index:Index
        }.

/* NP --> N2, RC */
parse_np(NP,H1,H3) -->
        { NP = syn:X,
          N2 = syn:X,
          RC = syn:X,
          NP = sem:in:Y,
          N2 = sem:in:Y,
          N2 = sem:out:Z,
          RC = sem:in:Z,
          RC = sem:out:W,
          NP = sem:out:W
        },
        parse_n2(N2,H1,H2),
        parse_rc(RC,H2,H3).

/* NP --> N2 */
parse_np(N2,H1,H2) --> parse_n2(N2,H1,H2).

/* N2 --> Det, N1 */
parse_n2(N2,H1,H2) -->
```

```
     { N1 = syn:class:common,
       N2 = syn:X,
       N1 = syn:X,
       Det = sem:Y,
       N2 = sem:Y,
       N1 = sem:Z,
       Det = sem:res:Z
     },
     parse_det(Det),
     parse_n1(N1,H1,H2).


/* N1 --> Adj, N1 */
parse_n1(N1,H1,H2) -->
     { N1 = syn:X,
       Adj = syn:X,
       N1a = syn:X,
       N1 = sem:in:Y,
       N1a = sem:in:Y,
       N1a = sem:out:Z,
       Adj = sem:in:Z,
       Adj = sem:out:W,
       N1 = sem:out:W
     },
     parse_adj(Adj),
     parse_n1(N1a,H1,H2).
```

```
/* N1 --> N */
parse_n1(N1,H,H) --> parse_noun(N1).

/* a rule for relative clauses */
parse_rc(RC,H1,H2) -->
     { RC = syn:index:Index,
       RC = sem:X,
       S = sem:X
     },
     ([who];[whom];[which];[that]),
     parse_sentence(S,[rel(Index)|H1],H2).

/* rules for verb phrases */
/* VP --> V, NP */
parse_vp(VP,H1,H2) -->
     { V = syn:class:transitive,
       VP = syn:X,
       V = syn:X,
       NP = sem:Y,
       VP = sem:Y,
       NP = syn:index:Z,
       VP = syn:arg2:Z,
       V = sem:W,
       NP = sem:scope:W
     },
     parse_verb(V),
     parse_np(NP,H1,H2).



/* VP --> V */
parse_vp(VP,H,H) -->
     parse_verb(VP),
     { VP = syn:class:intransitive }.

/* VP --> Adj */
parse_vp(VP,H,H) -->
     parse_adj(VP),
     { VP = syn:class:adjective,
       VP = syn:arg1:X,
       VP = syn:index:X
     }.

/* VP --> NP */
parse_vp(VP,H1,H2) -->
     { VP = sem:X,
       NP = sem:X,
       VP = syn:arg1:I1,
       NP = syn:index:I2,
       NP = sem:scope:in:[drs(U,C)|Sup_drs],
       NP = sem:scope:out:[drs(U,[(I1=I2)|C])|Sup_drs]
     },
     parse_np(NP, H1,H2),
     { VP = syn:class:common }.

/* lexical rules */



/* Aux --> [does] ; [is] */
parse_aux(_) --> [does] ; [is].

/* Det --> [a] ; [an] */
parse_det(Det) -->
     ([a] ; [an]),
     { Det = sem:in:X,
       Det = sem:res:in:X,
       Det = sem:res:out:Y,
       Det = sem:scope:in:Y,
       Det = sem:scope:out:Z,
       Det = sem:out:Z
     }.

/* Det --> [every] */
parse_det(Det) -->
     [every],
     { Det = sem:in:X,
       Det = sem:res:in:[drs([],[])|X],
       Det = sem:res:out:Y,
       Det = sem:scope:in:[drs([],[])|Y],
          Det = sem:scope:out:[Scope,Res,drs(U,C)|Sup_drs],
          Det = sem:out:[drs(U,[if_then(Res,Scope)|C])|Sup_drs]
        }.

/* N --> Proper_Noun */
parse_noun(N) -->
        [Word],
        { proper_noun(Word,I,Sem),
          add_topdrs(I,Sem,DRSlist,DRSlist1),
          N = syn:index:I,
          N = syn:class:proper,
          N = sem:in:DRSlist,
          N = sem:out:DRSlist1
        }.

add_topdrs(I,S,[drs(U,C)], [drs([I|U],NewC)]) :-
        append(S,C,NewC), !.
add_topdrs(I,S,[H|T],[H|L]):- add_topdrs(I,S,T,L).

/* Proper_Noun --> [john] */
proper_noun(john, X, [named(X,'John'), gender(X,m)]).
/* We can add more proper nouns hereafter. */

/* N --> Common_Noun */
parse_noun(N) -->
        [Word],
      { common_noun(Word,I,Sem),
        append(Sem,C,NewC),
        N = syn:index:I,
        N = syn:class:common,
        N = sem:in:[drs(U,C)|Sup_drs],
        N = sem:out:[drs([I|U],NewC)|Sup_drs]
      }.

/* Common_Noun --> [man] */
common_noun(man,X,[man(X),gender(X,m)] ).
common_noun(donkey,X,[donkey(X),gender(X,n)] ).
/* We can add more common nouns hereafter. */

/* Adj --> Word */
parse_adj(Adj) -->
      [Word],
      { adjective(Word,I,Sem),
        append(Sem,C,NewC),
        Adj = syn:index:I,
        Adj = sem:in:[drs(U,C)|Sup_drs],
        Adj = sem:out:[drs(U,NewC)|Sup_drs]
      }.

/* Word --> [old] */
adjective(old, X,[old(X)]).
/* We can add more adjectives hereafter. */
/* V --> Transitive */
parse_verb(V) -->
     [Word],
     { transitive_verb(Word,A1,A2,Sem),
       append(Sem,C,NewC),
       V = syn:class:transitive,
       V = syn:arg1:A1,
       V = syn:arg2:A2,
       V = sem:in:[drs(U,C)|Sup_drs],
       V = sem:out:[drs(U,NewC)|Sup_drs]
     }.

/* Transitive --> [owns] */
transitive_verb(owns, X,Y,[own(X,Y)]).
transitive_verb(own, X,Y,[own(X,Y)]).
/* We can add more transitive verbs hereafter. */


/* V --> Intransitive */
parse_verb(V) -->
     [Word],
     { intransitive_verb(Word,Arg,Sem),
       append(Sem,C,NewC),
       V = syn:class:intransitive,
       V = syn:arg1:Arg,
       V = sem:in:[drs(U,C)|Sup_drs],
    V = sem:out:[drs(U,NewC)|Sup_drs]
  }.

/* Intransitive --> [brays] */
intransitive_verb(brays, X,[bray(X)]).
intransitive_verb(bray, X,[bray(X)]).
/* We can add more intransitive verbs hereafter. */
```

REFERENCES

Allen, J. F. (1995). Natural Language Understanding. 2nd edition, Menlo Park, CA: Benjamin/Cummings.

Berwick, R. C. (1983). Computational Aspects of Discourse. In M. Brady and R. Berwick (eds.), Computational Models of Discourse (p.27-106). Cambridge, MA: MIT Press.

Berwick, R. C. (1989). Computational Linguistics. Cambridge, MA: MIT Press.

Bosch, P. (1983). Agreement and Anaphora: A Study of the Role of Pronouns in Syntax and Discourse. NY: Academic Press.

Chierchia, G. (1992). Anaphora and Dynamic Binding. Linguistics and Philosophy 15, 111-183.

Chomsky, N. (1981). Lectures on Government and Binding. Dordrecht: Foris.

Cooper, R. (1979). The Interpretation of Pronouns. In F. Heny and H. Schnelle (eds.), Syntax and Semantics 10. NY: Academic Press.

Covington, M. A. (1994). Natural Language Processing for Prolog Programmers. Englewood Cliffs, NJ: Prentice-Hall.

Covington, M. A. (1994). GULP 3.1: An Extension of Prolog for Unification-Based Grammar. ACMC Research Report 1994-06, University of Georgia.

Covington, M. A., Nute, D., Schmitz, N. and Goodman, D. (1988). From English to Prolog via Discourse Representation Theory. ACMC Research Report 01-0024, University of Georgia.

Covington, M. A. and Schmitz, N. (1988). An Implementation of Discourse Representation Theory. ACMC Research Report 01-0023, University of Georgia.

Evans, G. (1977). Pronouns, Quantifiers, and Relative Clauses. Canadian Journal of Philosophy 7, 467-536.

Evans, G. (1980). Pronouns. Linguistic Inquiry 11, 337-362.

Gazdar, G. and Mellish, C. (1989). Natural Language Processing in Prolog: An Introduction to Computational Linguistics. Reading, MA: Addison-Wesley.

Geach, P. (1972). Logic Matters. Oxford: Blackwell.

Grosz, J. (1986). The Representation and Use of Focus in a System for Understanding Dialogs. In B. J. Grosz, K. S. Jones and B. L. Webber (eds.), Reading in Natural Language Processing (p.353-362). Los Altos, CA: Kaufmann.

Heim, I. (1983). File Change Semantics and the Familiarity Theory of Definiteness. In R. Bauerle, C. Schwarze, and A. von Stechow (eds.), Meaning, Use, and Interpretation of Language (p. 164-198). Berlin: Walter de Gruyter.

Heim, I. (1990). E-Type Pronouns and Donkey Anaphora. Linguistics and Philosophy 13, 137- 178.

Hobbs, J. (1986). Resolving Pronoun References. In B. J. Grosz, K. S. Jones and B. L. Webber, (eds.), Reading in Natural Language Processing (p.353-362). Los Altos, CA: Kaufmann Publishers.

Johnson, M. and Klein, E. (1986). Discourse, Anaphora, and Parsing. CSLI Research Report 86-63, Stanford University.

Kadmon, N. (1990). Uniqueness. Linguistics and Philosophy 13, 272-324.

Kamp, H. (1981). A Theory of Truth and Semantic Representation. In J. Groenendijk, T. Janssen, and M. Stockhof (eds.), Truth, Interpretation, and Information (p.1-41). Dordecht: Foris.

Kaplan, J. (1983). Cooperative Responses from a Portable Natural Language Database Query System. In M. Brady and R. Berwick (eds.), Computational Models of Discourse (p.167-208). Cambridge, MA: MIT Press.

Kripke, S. (1972). Naming and Necessity. In D. Davidson and G. Harman (eds.), Semantics for Natural Language (p.253-355). Dordrecht: Reidel.

May, R. (1985). Logical Form: Its Structure and Derivation. Cambridge, Mass.: MIT Press.

May, R. (1989). Interpreting Logical Form. Linguistics and Philosophy 12, 387-435.

Neale, S. (1990). Descriptions. Cambridge. Mass: MIT Press.

Pereira, F.C.N. and B. Grosz (1994). Natural Language Processing. Cambridge, MA: MIT Press.

Reinhart, T. (1983). Anaphora and Semantic Interpretation. London: Croom Helm.

Roberts, C. (1989). Modal Subordination and Pronominal Anaphora in Discourse. Linguistics and Philosophy 12, 683-721

Russell, B. (1919). Descriptions. From Introduction to Mathematical Philosophy. London: G. Allen and Unwin Ltd.,, 167-180.

Schank, R. and R. Abelson. (1977). Scripts, Plans, Goals, and Understanding. Hillsdale, NJ: Lawrence Erlbaum Associates.

Sidner, C. (1983). Focusing in the Comprehension of Definite Anaphora. In M. Brady and R. Berwick (eds.), Computational Models of Discourse (p.267-330). Cambridge, MA: MIT Press.

Webber, B. L. (1983). So what can we talk about now? In M. Brady and R. Berwick (eds.), Computational Models of Discourse (p.331-370). Cambridge, MA: MIT Press.

Williams, E. (1977). Discourse and Logical Form. Linguistic Inquiry 8, 101-139.