Research Report AI–1994–03

A Natural Language Querying System Based on Discourse Representation Theory and Incorporating Event Semantics

Daniel Wayne Brown

Artificial Intelligence Center
The University of Georgia
Athens, Georgia 30602–7415 U.S.A.

Abstract

This thesis presents a querying system that accepts natural language inputs and is able to answer natural language queries about the inputs. The system works by translating English discourses into discourse representation structures, which are then transformed into Prolog facts and queries.

The program is an extension of a previous implementation of Discourse Representation Theory. Added functionality includes:

- asserting facts into and querying the Prolog database.
- handling the past tense.
- handling progressive forms.
- distinguishing between states and events.

The program was developed using GULP, an extension of Quintus Prolog that facilitates the implementation of unification–based grammars.

Acknowledgements

I would like to thank Dr. Michael Covington for suggesting corrections, giving me lots of LaTeX advice, and putting up with my infuriating tendency to procrastinate. Also, I am thankful to Drs. Donald Nute and Deborah Schmidt for their willingness to serve on my committee. I would also like to thank Laurel Graham for commiserating with me during times of heavy deadline pressure and driving on many a late–night trip to the convenience store. Finally, thanks are in order to Hans Kamp for coming out with a new book just when I needed it.

Contents

Chapter 1

Verb Semantics

1.1 Eventualities

The inspiration for this thesis was Terence Parsons' book, *Events in the Semantics of English* (1990). The book is a fleshing–out of Davidson's (1967) insights about the semantics of action verbs. Parsons adds details to deal with state verbs, tense and aspect, and a host of other features of natural languages. This thesis presents a querying system, DRT-E (Discourse Representation Theory with Eventualities), that takes Parsons' observations into account.

Davidson's theory was based on the notion that verbs introduce the existence of events, in much the same way that nouns introduce the existence of objects. Parsons traces this idea back to Frank Ramsey (1927), who stated:

> 'That Caesar died' is really an existential proposition, asserting the existence of an event of a certain sort.

Parsons gives several arguments for the underlying event analysis. One piece of evidence is that one can often give the same information with or without explicit reference to events. Parsons illustrates this with the following two sentences:

> After the singing of the *Marseillaise* they saluted the flag.
> After the *Marseillaise* was sung they saluted the flag.

The semantic content of the two sentences is basically identical,[1] though the

[1] The former sentence does seem, unlike the latter, to imply that there was only one singing of the *Marseillaise,* but this is not critical to the point being made.

former refers explicitly to an event, while the second does not.

Another bit of evidence comes from perceptual statements. Consider the sentence, "Mary saw Brutus stab Caesar." What exactly did Mary see? Did she see the *proposition* that Brutus stabbed Caesar? Parsons points out that this sentence says something very different from "Mary saw *that* Brutus stabbed Caesar." Perhaps Mary was not exactly sure what she saw. Perhaps she realized only that there was a struggle going on, and did not know the identity of either participant or that a stabbing was involved. This awkwardness is eliminated if what Mary saw was an event rather than a proposition; it is entirely plausible that one could see an event without recognizing its nature.

DRT-E uses the notion of underlying events in its internal representation of discourses. For example, consider the sentence:

Pedro sees Chiquita.

Ignoring events, the logical form of this sentence would be:

$$\exists x \exists y (\text{named}(x, \text{`Pedro'}) \land \text{named}(y, \text{`Chiquita'}) \land \text{sees}(x, y))$$

Taking events into account, the same sentence is partially interpreted as:

$$\exists e \exists x \exists y (\text{named}(x, \text{`Pedro'}) \land \text{named}(y, \text{`Chiquita'}) \land \text{seeing}(e, x, y))$$

which might be paraphrased as:

> There exists a seeing event such that the subject of the seeing is named Pedro and the object of the seeing is named Chiquita.[2]

1.2 *Aktionsarten*

Vendler (1957) introduced a fourfold semantic classification of verbs into 'activities,' 'accomplishments,' 'achievements,' and 'states.' These classes have come to be known as '*Aktionsarten,*' which is German for 'kinds of actions.'[3]

A *state* verb expresses a quality that is true either timelessly or for a specified period of time. An example is 'loves' in "Pedro loves Chiquita."

[2]Explicit representation of thematic roles was deemed to be tangential to the purpose of this project.

[3]The term was first used by the *Junggrammatiker* (Neogrammarians), a mostly German school of linguists that flourished around the turn of the century (Kamp and Reyle 1993).

State verbs are not used in the present progressive (e.g., "Pedro is loving Chiquita" is awkward). Adjectives also express states.

An *achievement* verb[4] refers to an instantaneous event. An example is 'slapped' in "Pedro slapped Chiquita." It does not make sense to ask when an achievement started or how long it took.

An *activity* verb relates a non–instantaneous event. An example is 'fought' in "Pedro fought Chiquita." It is appropriate to ask the *starting* and *stopping* times of an activity.

An *accomplishment* verb expresses an activity ending with an achievement. An example is 'ate' in "Pedro ate the haggis." (Here the activity is Pedro's eating process; the achievement is his completion of the eating.) It is appropriate to ask the *finishing* time of an accomplishment.

(It is worth noting that verbs can often be extended for use as members of different classes. For example, even though 'find' is generally considered an achievement verb, it is acceptably used as an accomplishment verb in the sentence, "How long did it take to find it?")

It turns out that the *Aktionsart* of a verb plays a significant role in finding suitable representations for English sentences in which it occurs. The most significant distinction is between states and the other three classes, which henceforth will be collectively referred to as 'events.' The union of these two superclasses will be termed 'eventualities.'[5]

One of the most obvious manifestations of the distinction among verb classes is that some verbs have continuous forms, while others do not. Vendler points out that in English, only event verbs can be used to answer the question, "What are you doing?" Valid answers include "I am {running, working, eating, etc.}" but not "I am {knowing, loving, seeing}." On the other hand, only state verbs can be used in the question, "Do you ⟨*verb*⟩?" (if the question is taken to mean that the eventuality applies to a present rather than a habitual action).

The most important distinction between events and states for the purpose of constructing a model is that events have culmination points (Parsons 1990), i.e., times at which they finish. States can merely be said to hold or not at a given time; it is awkward to say that a state is 'finished'. DRT-E reflects this by means of the special predicates *cul* and *hold.*

DRT-E also has a third reserved predicate, *hab,* which is used to handle habitual readings, most notably for the simple present. This predicate is

[4]Fleck (1988) prefers the term 'state change.'

[5]This usage is consistent with that of Parsons (1990).

needed for the correct handling of queries about habitual action.

1.3 Tense

The handling of time in DRT-E is inspired by Reichenbach's (1947) system of tense logic. Reichenbach described tenses in relation to three times: time of occurrence, time of utterance, and time of reference. The first two are self–explanatory. The reference time is the time that the utterer is talking about.

Reichenbach's system is good at explaining exactly what different tenses mean. The present tense indicates that all three times are concurrent. The past tense indicates that the time of occurrence is concurrent with the time of reference, which is before the time of utterance.

The time of reference becomes significant in cases where the tense expresses a relation to a time other than the utterance time. An example is the past perfect form, which indicates that the reference time is before the utterance time but after the occurrence time. Since the current implementation does not handle perfect forms, it does not take reference time into account.

Chapter 2

Discourse Representation Theory

2.1 Model Theory

DRT-E uses Discourse Representation Theory (Kamp 1981; Kamp and Reyle 1993), commonly known as DRT, as an interlingua between the English inputs provided by the end user and their Prolog representations. DRT is a method of representing a discourse composed of multiple sentences with a logical form that captures the truth conditions of the discourse.

DRT is based on a model–theoretic semantics. In DRT, the truth conditions for a discourse depend on how it fits into some model.

A model is a precisely defined knowledge base. A model for a language is defined as an ordered pair (**D**, **I**), where **D** is the domain (i.e. the set of things that can be referred to in the model) and **I** is an interpretation function that maps from the language to the domain.

I maps logical constants[1] onto elements of **D** and maps predicates onto sets of tuples of elements of **D**. For example, in the model of the current implementation, $\mathbf{I}(beats)$ is the set of all tuples $\langle x, y\rangle$ such that x beats y.

Model theory makes it possible to define precisely what is means for a formula to be true. In particular, $f(x_1, x_2, \ldots)$ is true if and only if $\langle \mathbf{I}(x_1), \mathbf{I}(x_2), \ldots\rangle \in \mathbf{I}(f)$. The truth value of a formula constructed by appending a unary logical connective to another formula or joining two formulas with a binary logical connective is determined according to the truth

[1]There are no logical constants in the current implementation. Even proper names are treated as properties of individuals, not as unique identifiers.

value(s) of the formula(s) and the definition of the connective. Some examples:

- $\neg P$ is true if and only if P is false. true.
- $P \rightarrow Q$ is true if and only if P is false or Q is true (or both). of x in **D**.
- $\exists x P$ is true if and only if P is true for at least one value of x in **D**.

2.2 Discourse Representation Structures

In DRT, a discourse is represented by a discourse representation structure (DRS). A DRS consists of an ordered pair (U, Con), where U is a set of entities in the universe of discourse, and Con is a set of conditions that constrains the relationships of those entities. In other words, U corresponds to the domain of a model, and the interpretation of Con in the model determines the truth conditions of the discourse represented by a DRS.

A discourse is true in a given model if and only if its DRS can be embedded in the model in such a way that all the conditions are satisfied. In other words, there must be a way to instantiate each discourse referent to a member of the model's domain so that each of the discourse conditions is satisfied by the model's interpretation.

For example, the discourse

Pedro owns a green donkey. He beats it.

can be naively represented by the following DRS: $(\{x, y, z, w\}$, $\{\text{named}(x, \text{‘Pedro’}), \text{donkey}(y), \text{green}(y), \text{owns}(x, y), z = x, y = w, \text{beats}(z, w)\})$ or in diagrammatic form:

x, y, z, w
named(x,‘Pedro’) donkey(y) green(y) owns(x, y) $z = x$ $y = w$ beats(z, w)

In practice, the equivalent discourse referents are combined, so the above discourse becomes

$$\begin{array}{|l|}\hline x, y \\ \hline \text{named}(x,\text{`Pedro'}) \\ \text{donkey}(y) \\ \text{green}(y) \\ \text{owns}(x, y) \\ \text{beats}(x, y) \\ \\ \hline \end{array}$$

The first–order logic equivalent of this DRS is

$$\exists x \exists y (\text{named}(x, \text{`Pedro'}) \wedge \text{donkey}(y) \wedge \text{green}(y) \wedge \text{beats}(x, y))$$

Therefore the discourse is true if and only if there exist x and y such that x is named Pedro, y is a donkey, y is green, x owns y, and x beats y.

Anaphoric pronouns, like 'he' and 'it' in the above example, are rather tricky to resolve. There is no completely satisfactory way to determine what an anaphoric pronoun refers to. DRT-E simply resolves a pronoun with the most recent noun that agrees with it in gender.[2]

Conditional statements are a bit more complicated. The antecedent and consequent parts each have their own sub–DRSes. For example, the sentence

If Pedro is hungry, he eats.

can be represented as:[3]

$$\begin{array}{|c|}\hline \\ \hline \begin{array}{|l|}\hline x \\ \hline \text{named}(x,\text{`Pedro'}) \\ \text{hungry}(x) \\ \\ \hline \end{array} \quad \Rightarrow \quad \begin{array}{|l|}\hline \\ \hline \text{eats}(x) \\ \hline \end{array} \\ \\ \hline \end{array}$$

[2]Since plurals have not yet been implemented, number agreement is not enforced for pronouns. See Izzo (1993) and Graham (forthcoming) for DRT implementations that handle plurals.

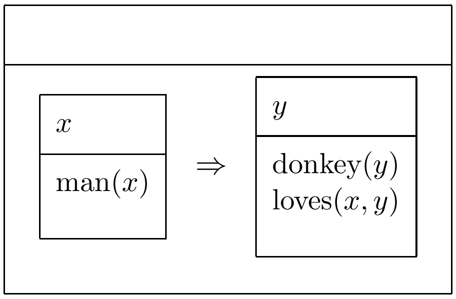
[3]Actually, proper names should be raised to the outermost DRS, since they should be accessible throughout the discourse. Unfortunately, this has not been done in the current implementation.

The sub–DRSes connected by the arrow represent a DRS–condition that is satisfied if and only if every value of x that makes the left–hand sub–DRS true makes the right–hand one true also.

Universal statements receive similar treatment. The sentence

Every man loves a donkey.

is represented as:

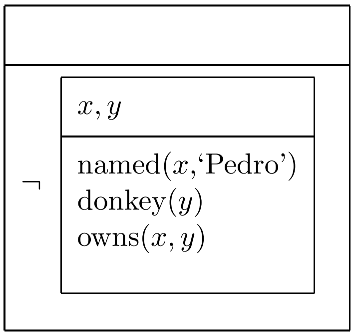


The sub–DRS to the left of the arrow denotes the scope of the conditional, while the one to the right is the restrictor. Consequently, the discourse referents in the left DRS are available for use in the right one, though the reverse is not true.

Negative sentences are represented by negated DRSes. For example, the sentence

Pedro does not own a donkey.

is represented as:

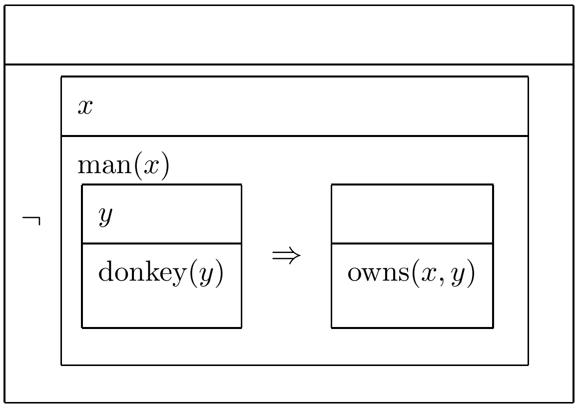


The condition expressed by the negated DRS is satisfied if and only if there is no assignment of values to x and y that satisfies the DRS without the negation.

Negative existential sentences combine the features of the two types of DRS. An example is the sentence

No man owns a donkey.

which is represented by



2.3 Temporal Features

The version of DRT used in DRT-E goes into some further detail on the semantics of verbs. DRS construction takes into account the theory of underlying events; it also distinguishes between states and events and handles some temporal and aspectual information.

Since verbs introduce the existence of events, it is necessary to add a discourse referent whenever a new event is described. That referent can then be used when noting characteristics of the event — most notably, the time of the event.

The representation of the time of an event varies according to its *Aktionsart*, tense, and aspect. The *cul, hold,* and *hab* predicates are used to associate an eventuality with a time. State verbs always utilize the *hold* predicate. Event verbs may use any of the three, depending on how they are used.

A description of an event that utilizes the simple past tense is assumed to be a reportive use and therefore requires the *cul* predicate. For example, the sentence

Pedro fed Chiquita.

would have as part of its semantics $cul(e, t)$, where e is the event marker for Pedro's feeding of Chiquita (his donkey) and t is the time at which the event culminated.

Progressive forms of events require the use of the *hold* predicate. A progressive sentence, such as

Pedro is feeding Chiquita.

describes a state. Parsons (1990) calls such a state the 'in–process state' of its corresponding event. The above sentence describes the in–process state of a feeding event. Parsons suggests that an 'in–process' predicate be introduced to distinguish the state from the event. The current implementation, however does not do this; using *hold* instead of *cul* is sufficient distinction for the purposes of the program.

The *hab* predicate is used to handle the habitual sense of a verb. DRT-E assumes that an instance of the simple present form of an event verb indicates a habitual usage; e.g. "Pedro feeds Chiquita" is taken to mean that it is a normal practice for Pedro to feed Chiquita.

Time itself is handled by introducing a special discourse referent *now* as part of every discourse and expressing the relationship of the time of an eventuality to *now* (Kamp and Reyle 1993). The special predicates *at* and *before* are used to express the relationship between two times. These two are sufficient, since DRT-E currently handles only the present and past tenses.

An example of a DRS for a present tense state sentence is:

Pedro sees Chiquita.

s, x, y, t, now
named(x,'Pedro') named(y,'Chiquita') seeing(s, x, y) hold(s, t) at(t,now)

Paraphrased, the DRS states that there is a seeing–state that holds now that has Pedro as its subject and Chiquita as its object.

An example of a DRS for a past tense event sentence is:

Pedro fed Chiquita.

e, x, y, t, now
named(x,'Pedro')
named(y,'Chiquita')
feeding(e, x, y)
cul(e, t)
before(t,now)

The DRS indicates that there was a feeding–event that culminated before now that had Pedro as its subject and Chiquita as its object.

The habitual case is handled analogously:

Pedro beats Chiquita.

s, x, y, t, now
named(x,'Pedro')
named(y,'Chiquita')
beating(s, x, y)
hab(s, t)
at(t,now)

The best way to explain this DRS is to say that Pedro's beating of Chiquita is a habitual state that holds now, even though an actual beating–event may not be taking place at the moment.

Progressive forms make event verbs refer to states, e.g.:

Pedro was eating.

s, x, t, now
named(x,'Pedro')
eating(s, x)
hold(s, t)
before(t,now)

The use of *hold* instead of *cul* makes it clear that the in–process state of an eating event, rather than the event itself, is involved here.

Adjectives also introduce states:

Pedro is happy.

$$
\begin{array}{|l|}
\hline
s, x, t, now \\
\hline
\text{named}(x,\text{'Pedro'}) \\
\text{happy}(s, x) \\
\text{hold}(s, t) \\
\text{at}(t, now) \\
\\
\hline
\end{array}
$$

2.4 Query Processing

DRT-E represents queries with an *ad hoc* representation that is not part of DRT. An interrogative sentence introduces a sub–DRS labeled as a query. For example, the query

Is Pedro beating Chiquita?

can be represented as:

$$
\begin{array}{|l|}
\hline
\\
\hline
?: \begin{array}{|l|}
\hline
s, x, y, t, now \\
\hline
\text{named}(x,\text{'Pedro'}) \\
\text{named}(y,\text{'Chiquita'}) \\
\text{beating}(s, x, y) \\
\text{hold}(s, t) \\
\text{at}(t, now) \\
\\
\hline
\end{array} \\
\hline
\end{array}
$$

Unlike the other kinds of sub–DRSes introduced previously, query DRSes do not have truth values. They do not affect the truth of superordinate DRSes in any way. Their purpose is dual: to provide a representation for interrogative sentences, and to aid in the construction of database queries.

There are two classes of queries that need some extra processing in the current implementation. One class includes queries of the form, “Does $\langle NP \rangle$ $\langle VP \rangle$?” The other class includes queries of the form, “Did $\langle NP \rangle$ $\langle VP \rangle$?”

Does–questions require special treatment when the verb phrase describes an event. Since the simple present form of an event verb indicates a habitual usage, such a query is assumed to be asking about a habitual occurrence. Consequently, any instances of *cul* in the query conditions must be replaced with *hab.* For example,

Does Pedro beat Chiquita?

is represented as:

$$
\begin{array}{|l|}
\hline
\quad \\
\hline
?: \begin{array}{|l|}
\hline
s, x, y, t, now \\
\hline
\text{named}(x,\text{‘Pedro’}) \\
\text{named}(y,\text{‘Chiquita’}) \\
\text{beating}(s, x, y) \\
\text{hab}(s, t) \\
\text{at}(t, now) \\
\\
\hline
\end{array} \\
\hline
\end{array}
$$

Did–questions are ambiguous. They can ask about an event that occurred in the past, a state that held in the past, or an eventuality that was common in the past. Therefore all uses of *cul, hold,* and *hab* in such a query are replaced with the disjunction of all three. For example,

Did Pedro beat Chiquita?

is represented as:

$$
\begin{array}{|l|}
\hline
\\
\hline
?: \quad \begin{array}{|l|}
\hline
s, x, y, t, now \\
\hline
\text{named}(x,\text{`Pedro'}) \\
\text{named}(y,\text{`Chiquita'}) \\
\text{beating}(s, x, y) \\
\text{cul}(s, t) \vee \text{hold}(s, t) \vee \text{hab}(s, t) \\
\text{before}(t, now) \\
\\
\hline
\end{array} \\
\\
\hline
\end{array}
$$

Chapter 3

Implementation

3.1 Building DRSes

DRT-E's DRS–building algorithm is based on that of Johnson and Klein (1986). See also Covington and Schmitz (1988) and Goodman (1989), which discuss previous versions of the program, for a more detailed treatment.[1]

Discourse structure is determined by means of a simple top–down parser written in definite clause grammar (DCG) notation. The parser implements a unification–based grammar that enforces subject–verb agreement[2] and builds discourse representation structures.

A DRS is represented in Prolog as `drs(U,Con)`, where `U` is a list of the discourse referents in the DRS, and `Con` is a list of predicates that indicate the conditions that constrain the elements in the discourse. For example, the DRS for

Pedro is eating.

which is shown above in graphical form, becomes in Prolog something like

```
drs([now,_7410,_7546,_7548],[at(_7548,now),hold(_7546,_7548),
     eating(_7546,_7410),named(_7410,Pedro)])
```

(The underscores followed by numbers are variable names generated by Prolog.)

[1] This section closely follows the Covington and Schmitz paper.

[2] Even though plural nouns have not been implemented yet, there are cases where the check is needed to disambiguate the verb form; e.g. 'beat' is both the simple past and the plural present of the verb.

Most of the work of the parser is done by feature structure unification. DRT-E was developed using GULP (Graph Unification Logic Programming) 3.0a (Covington 1993), an extension of Quintus Prolog that facilitates the implementation of unification–based grammars. The first argument of each node in the parse tree is a feature structure of the form:

$$\left[\begin{array}{ll}
\textit{syn:} & \left[\begin{array}{l} \textit{index: } \ldots \\ \textit{class: } \ldots \\ \textit{akt: } \ldots \\ \textit{arg1: } \ldots \\ \textit{arg2: } \ldots \end{array}\right] \\
\textit{sem:} & \left[\begin{array}{ll} \textit{in: } \ldots & \\ \textit{out: } \ldots & \\ \textit{res:} & \left[\begin{array}{l} \textit{in: } \ldots \\ \textit{out: } \ldots \end{array}\right] \\ \textit{scope:} & \left[\begin{array}{l} \textit{in: } \ldots \\ \textit{out: } \ldots \end{array}\right] \end{array}\right] \\
\textit{agr:} & \left[\begin{array}{l} \textit{num: } \ldots \\ \textit{tense: } \ldots \\ \textit{asp: } \ldots \end{array}\right]
\end{array}\right]$$

The roles of the features are as follows:

index The discourse marker associated with the node. Instantiated only on nouns and nodes that dominate, modify (e.g. adjectives), or unify with nouns.

class Instantiated to *common* or *proper* on nouns; instantiated to *transitive* or *intransitive* on verbs.

akt *Aktionsart* of the verb.

arg1, arg2 Discourse markers for subject and direct object of the verb.

sem:in State of DRS before processing the current node.

sem:out State of DRS after processing the current node.

res, scope Used to determine the logical structure of a sentence.

agr Contains information on tense, aspect, and number for the enforcement of subject–verb agreement.

To illustrate, this is the phrase–structure rule for common nouns:

```
n(N) --> [Form],
         { common_noun(Form,Num,lambda(I,Semantics)),
           append(Semantics,Con,NewCon),
           N = syn: (index:I ::
                     class:common) ::
               sem: (in:  [drs(U,Con)|Super] ::
                     out: [drs([I|U],NewCon)|Super]) ::
               agr: num:Num }.
```

Besides accepting a noun, this rule accomplishes several things:

- Generates a new variable (I), which is later instantiated to become a unique discourse marker for the noun.
- Determines the number of the noun so that subject–verb agreement can be enforced.
- Adds the index of the noun to U and the semantics of the noun to *Con* of the current DRS.

Nouns are stored in a table that includes the form, number, and semantics of each noun. A sample entry is

```
common_noun(donkey,sg,lambda(X,[donkey(X),gender(X,n)])).
```

The first argument is the actual form of the noun. The second indicates that the noun is singular. The third expresses the semantics of the noun; in this case, it indicates that 'donkey' introduces a discourse referent X such that X is a donkey and the gender of X is neuter.[3]

Verb rules are similar. This is the rule for simple present intransitive verbs:

[3] The gender eventually gets discarded, as it is not really part of the semantics of the sentence; it is only used in determining the antecedents of pronouns.

```
v(V) --> [Form],
         { verb(Form,Akt,Num,Tense,Asp,
                lambda(I,Arg,Time,Semantics)),
           append(Semantics,Con,NewCon),
           V = syn : (class:intransitive ::
                      akt:Akt ::
                      arg1:Arg) ::
               sem : (in:  [drs(U,Con)|Super] ::
                      out: [drs([Time,I|U],NewCon)|Super]) ::
               agr: (num:Num ::
                     tense:Tense ::
                     asp:Asp) }.
```

The explanation is much the same for verbs as for nouns, except that there are a few more features that have to be instantiated for verbs, and an extra discourse referent is introduced for the occurrence time.

A sample lexical entry for a verb is as follows:

```
verb(sees,s,sg,pres,_,
     lambda(S,X,Y,T,[at(T,now),hold(S,T),seeing(S,X,Y)])).
```

Again, the first argument is the word itself. The second indicates that 'sees' is a state verb. The third and fourth arguments show that the form of the verb is singular and present tense. The fourth argument indicates that the aspect will remain uninstantiated. The fifth means that 'sees' introduces discourse referents S, X, Y, and T such that S is a seeing–event with subject X and object Y, S holds at time T, and T is concurrent with *now*.

3.2 Translating DRSes Into Prolog

For the simplest DRSes, translating into Prolog is almost trivial. Discourse conditions are already in the form of Prolog clauses before the translation process begins.

Simple queries are the easiest to handle. The conditions are in the exact form needed for queries. For example,

Does Pedro love Chiquita?

which is represented by the DRS

$$
\begin{array}{|l|}
\hline
\\
\hline
?: \begin{array}{|l|}
\hline
s, x, y, t, now \\
\hline
\text{named}(x,\text{`Pedro'}) \\
\text{named}(y,\text{`Chiquita'}) \\
\text{loving}(s, x, y) \\
\text{hold}(s, t) \\
\text{at}(t, now) \\
\\
\hline
\end{array} \\
\\
\hline
\end{array}
$$

becomes the Prolog query:

```
?- named(X,pedro), named(Y,chiquita), loving(S,X,Y), hold(S,T),
   at(T,now).
```

Simple statements require a bit more tinkering. Simply asserting the conditions will not work, as there is an implied existential quantifier, while Prolog assumes an implied *universal* quantifier. Asserting something like `named(X,pedro)` would result in everything being named Pedro. The desired effect is to note the existence of just one thing named Pedro.

Since Prolog has no way of representing existential quantifiers, they must be eliminated somehow. This is done by a process known as skolemization (Skolem 1928), in which existentially quantified variables are replaced with unique identifiers. The current implementation uses lists of numbers for this purpose. In simple cases, a one element list is sufficient. For example,

Pedro loves Chiquita.

is represented by the DRS

$$
\begin{array}{|l|}
\hline
s, x, y, t, now \\
\hline
\text{named}(x,\text{`Pedro'}) \\
\text{named}(y,\text{`Chiquita'}) \\
\text{loving}(s, x, y) \\
\text{hold}(s, t) \\
\text{at}(t, now) \\
\\
\hline
\end{array}
$$

which is asserted into the Prolog database as something like

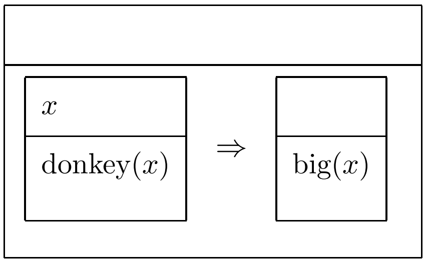
```
named([1],pedro).
named([2],chiquita).
loving([3],[1],[2]).
hold([3],[4]).
at([4],now).
```

The above query, "Does Pedro love Chiquita?" should be answered affirmatively after this statement has been processed, because there is in the knowledge base a loving–event (`[3]`) with Pedro (`[1]`) as its subject and Chiquita (`[2]`) as its object.

Simple if–then conditions are easy to handle. An if–then DRS basically corresponds to a Prolog rule. For example,

Every donkey is big.

which is represented as:[4]



is asserted as the rule

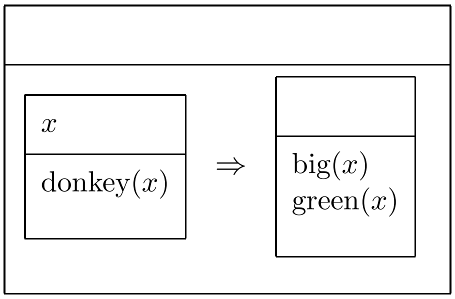
```
big(X) :- donkey(X).
```

A problem arises when the right–hand side of an if–then DRS has more than one condition because a Prolog rule cannot have more than one consequent. The current implementation gets around this problem by providing an operator `::-` that is the same as `:-` except that it allows compound consequents and can be queried. This operator is only used in intermediate processing, however; consequents are broken up before rules are asserted into the database. For example,

Every donkey is big and green.

which has the DRS

[4] Temporal information is not provided here because statements of equality are assumed to be timeless in the current implementation.



is initially translated as:

```
big(X), green(X) ::- donkey(X).
```

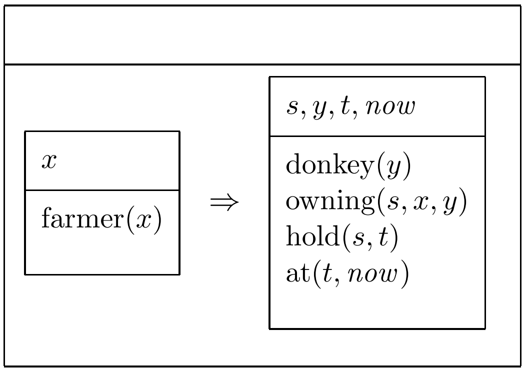
but is asserted as:

```
big(X) :- donkey(X).
green(X) :- donkey(X).
```

Things get complicated when discourse referents are introduced in the consequent of an if–then DRS. Consider the sentence

Every farmer owns a donkey.

which is represented as:



Obviously,

```
donkey(Y), owning(S,X,Y), hold(S,T), at(T,now) ::- farmer(X).
```

will not do, because that means that every farmer owns *every* donkey. Also,

```
donkey([1]), owning(S,X,[1]), hold(S,T), at(T,now) ::-
       farmer(X).
```

won't work, because that means that every farmer owns *the same* donkey (`[1]`).

The trick is to make the identity of the donkey dependent on the identity of the farmer. This is done in the current implementation by giving the donkey a dummy name that includes the farmer's, viz.:

```
donkey([1|X]), owning(S,X,[1|X]), hold(S,T), at(T,now) ::-
       farmer(X).
```

This approach distinguishes among donkeys owned by different farmers.[5]

DRT-E does not fully support negative statements and queries at this time. Prolog lacks a facility for expressing negation explicitly, which is necessary for handling negative queries properly. See Izzo (1993) for a similar implementation that handles negation.

3.3 Queries

When given a natural language discourse, DRT-E outputs the corresponding DRS, a record of database assertions and queries, and the results of the queries. A simple example:

```
|: Pedro is happy.  Is he happy?

[pedro,is,happy,.,is,he,happy,?]
--------------------------------------------------
[now,_9962,_9989,_9990]
QUERY:
  [_10279,_10280]
  happy(_10279,_9962)
  hold(_10279,_10280)
  at(_10280,now)
at(_9990,now)
hold(_9989,_9990)
happy(_9989,_9962)
named(_9962,Pedro)
--------------------------------------------------
Asserting: named([6],Pedro)
```

[5]In effect, `[1|X]` is a Skolem function (Skolem 1928) of `X`, because it is uniquely instantiated for every different value of `X`.

```
Asserting: happy([5],[6])
Asserting: hold([5],[4])
Asserting: at([4],now)
Querying:  happy(_10279,[6]),hold(_10279,_10280),at(_10280,now)
Result:    yes
```

The first line of output is the result of transforming the discourse into a list of Prolog atoms. The part between the dashed lines is the resulting DRS displayed in indented form. The remainder tells what database transactions occurred.

The first line of the DRS is its universe of discourse. The remainder is composed of DRS conditions and sub–DRSes. In the above example, the only sub–DRS is the query, which is represented by the marker `QUERY:` followed by four indented lines. The conditions come out in reverse order; notice that the query in the example comes before the assertions. However, as the database record shows, the processing is done in the correct order.

A more complicated example:

```
|: Pedro is green.  If Pedro is green then he is happy.  Is he
happy?

[pedro,is,green,.,if,pedro,is,green,then,he,is,happy,.,is,he,
 happy,?]
-------------------------------------------------
[now,_9036,_9063,_9064]
QUERY:
  [_9957,_9958]
  happy(_9957,_9036)
  hold(_9957,_9958)
  at(_9958,now)
IF:
  [_9501,_9502]
  green(_9501,_9036)
  hold(_9501,_9502)
  at(_9502,now)
THEN:
  [_9669,_9670]
  happy(_9669,_9036)
  hold(_9669,_9670)
  at(_9670,now)
```

```
at(_9064,now)
hold(_9063,_9064)
green(_9063,_9036)
named(_9036,Pedro)
--------------------------------------------------
Asserting: named([2],Pedro)
Asserting: green([1],[2])
Asserting: hold([1],[0])
Asserting: at([0],now)
Asserting: happy([4,_9502,_9501],[2]):-green(_9501,[2]),
           hold(_9501,_9502),at(_9502,now)
Asserting: hold([4,_9502,_9501],[3,_9502,_9501]):-
             green(_9501,[2]),hold(_9501,_9502),at(_9502,now)
Asserting: at([3,_9502,_9501],now):-green(_9501,[2]),
           hold(_9501,_9502),at(_9502,now)
Querying:  happy(_9957,[2]),hold(_9957,_9958),at(_9958,now)
Result:    yes
```

In this case there are two sub–DRSes: the query and the if–then DRS.

This is an example of how questions beginning with 'Does' are answered:

```
|: Pedro beats Chiquita.  Does he beat her?

[pedro,beats,chiquita,.,does,he,beat,her,?]
--------------------------------------------------
[now,_11611,_11824,_11834,_11837]
QUERY:
  [_12372,_12375]
  beating(_12372,_11611,_11824)
  hab(_12372,_12375)
  at(_12375,now)
at(_11837,now)
hab(_11834,_11837)
beating(_11834,_11611,_11824)
named(_11824,Chiquita)
named(_11611,Pedro)
--------------------------------------------------
Asserting: named([8],Pedro)
Asserting: named([7],Chiquita)
Asserting: beating([6],[8],[7])
```

```
Asserting: hab([6],[5])
Asserting: at([5],now)
Querying:  beating(_12372,[8],[7]),hab(_12372,_12375),
           at(_12375,now)
Result:    yes
```

Note that the query is assumed to be asking about a habitual action. That is why the following query fails:

```
|: Pedro is beating Chiquita.  Does he beat her?

[pedro,is,beating,chiquita,.,does,he,beat,her,?]
--------------------------------------------------
[now,_13428,_13641,_13651,_13654]
QUERY:
  [_14190,_14193]
  beating(_14190,_13428,_13641)
  hab(_14190,_14193)
  at(_14193,now)
at(_13654,now)
hold(_13651,_13654)
beating(_13651,_13428,_13641)
named(_13641,Chiquita)
named(_13428,Pedro)
--------------------------------------------------
Asserting: named([12],Pedro)
Asserting: named([11],Chiquita)
Asserting: beating([10],[12],[11])
Asserting: hold([10],[9])
Asserting: at([9],now)
Querying:  beating(_14190,[12],[11]),hab(_14190,_14193),
           at(_14193,now)
Result:    no
```

Contrast with the appropriate question:

```
|: Pedro is beating Chiquita.  Is he beating her?
[pedro,is,beating,chiquita,.,is,he,beating,her,?]
--------------------------------------------------
[now,_8884,_9097,_9107,_9110]
```

```
QUERY:
  [_9644,_9647]
  beating(_9644,_8884,_9097)
  hold(_9644,_9647)
  at(_9647,now)
at(_9110,now)
hold(_9107,_9110)
beating(_9107,_8884,_9097)
named(_9097,Chiquita)
named(_8884,Pedro)
--------------------------------------------------
Asserting: named([3],Pedro)
Asserting: named([2],Chiquita)
Asserting: beating([1],[3],[2])
Asserting: hold([1],[0])
Asserting: at([0],now)
Querying:  beating(_9644,[3],[2]),hold(_9644,_9647),
           at(_9647,now)
Result:    yes
```

The following examples illustrate the handling of questions beginning with 'Did.' The first is a simple past reading:

```
|: Pedro beat Chiquita.  Did he beat her?

[pedro,beat,chiquita,.,did,he,beat,her,?]
--------------------------------------------------
[now,_15177,_15390,_15400,_15403]
QUERY:
  [_16994,_16997]
  beating(_16994,_15177,_15390)
  cul(_16994,_16997);hold(_16994,_16997);hab(_16994,_16997)
  before(_16997,now)
before(_15403,now)
cul(_15400,_15403)
beating(_15400,_15177,_15390)
named(_15390,Chiquita)
named(_15177,Pedro)
--------------------------------------------------
Asserting: named([16],Pedro)
```

```
Asserting: named([15],Chiquita)
Asserting: beating([14],[16],[15])
Asserting: cul([14],[13])
Asserting: before([13],now)
Querying:  beating(_16994,[16],[15]),(cul(_16994,_16997);
             hold(_16994,_16997);hab(_16994,_16997)),
           before(_16997,now)
Result:    yes
```

The second is a progressive reading:

```
|: Pedro was beating Chiquita.  Did he beat her?

[pedro,was,beating,chiquita,.,did,he,beat,her,?]
--------------------------------------------------
[now,_18252,_18465,_18475,_18478]
QUERY:
  [_19014,_19017]
  beating(_19014,_18252,_18465)
  cul(_19014,_19017);hold(_19014,_19017);hab(_19014,_19017)
  before(_19017,now)
before(_18478,now)
hold(_18475,_18478)
beating(_18475,_18252,_18465)
named(_18465,Chiquita)
named(_18252,Pedro)
--------------------------------------------------
Asserting: named([20],Pedro)
Asserting: named([19],Chiquita)
Asserting: beating([18],[20],[19])
Asserting: hold([18],[17])
Asserting: before([17],now)
Querying:  beating(_19014,[20],[19]),(cul(_19014,_19017);
             hold(_19014,_19017);hab(_19014,_19017)),
           before(_19017,now)
Result:    yes
```

The habitual interpretation cannot be shown, since DRT-E does not currently recognize past habitual actions.

Chapter 4

Further Research Possibilities

The most obvious possibility for further extension of DRT-E would be to integrate it with the implementations of Izzo (1993) and Graham (forthcoming). Izzo's implementation incorporates defeasible reasoning, explicit negativity, and generic plurals. Graham's implementation adds a limited capability to handle plurals and definite descriptions.

The past tense of the copula ('was'/'were') has not been implemented because doing so would involve a new representation for equivalence. Specifically, equivalence needs to be made relative to time. This could be done in part by representing equality similarly to other predicates — i.e. making a three–place predicate with an associated holding time. However, some special inference procedures would have to be added for this predicate. This extension would become even more useful if implemented in conjunction with a table of identity — that is, a table that lists pairs of items that should be unifiable even though they have different discourse referents — so that queries about an item would have the same truth conditions regardless of what name is used to refer to it.

The past habitual reading should be fairly easy to add. The construction 'used to' marks this reading. All that is really needed are a couple of new verb rules. Queries beginning with 'Did' already result in affirmative answers when they refer to past habitual actions. Optionally, one could introduce queries containing 'use to' (e.g., "Did Pedro use to beat Chiquita?"), which are acceptable in some English dialects.

Other tenses would not be difficult to add. The future could be interpreted in much the same way as the past, using *after* instead of *before*. Perfect tenses could be handled if reference time were taken into account.

Another useful extension would be a facility for queries about the temporal relationship between two events, e.g. whether one event happened before another. Ideally, this would include default reasoning about the sequence of eventualities in a discourse. For example, the program should recognize that if two past events are mentioned consecutively, the one that was mentioned first probably happened first.

Bibliography

Covington, Michael (1993) *Brief notes on GULP 3.* Artificial Intelligence Programs, University of Georgia.

Covington, Michael, and Schmitz, Nora (1988) *An implementation of discourse representation theory.* ACMC Research Report 01–0023, University of Georgia.

Covington, Michael; Nute, Donald; Schmitz, Nora; and Goodman, David (1988) *From English to Prolog via discourse representation theory.* ACMC Research Report 01–0024, University of Georgia.

Davidson, Donald (1967) The logical form of action sentences. In N. Rescher, ed., *The logic of decision and action* 81–95. Pittsburgh: University of Pittsburgh Press.

Fleck, Margaret (1988) *Boundaries and topological algorithms.* Technical Report 1065, MIT Artificial Intelligence Laboratory.

Goodman, David (1989) *An implementation of an extension to discourse representation theory: translating natural language to discourse representation structures to Prolog clauses.* Master's Thesis, University of Georgia.

Graham, Laurel (forthcoming) *An implementation of plurality in discourse representation theory.* Master's Thesis, University of Georgia.

Izzo, Gregory John (1993) *Incorporating defeasible reasoning into an implementation of discourse representation theory.* Master's Thesis, University of Georgia.

Johnson, Mark, and Klein, Ewan (1986) *Discourse, anaphora, and parsing.* CSLI Research Report 86–63, Stanford University.

Kamp, Hans (1981) A theory of truth and semantic representation. In J. Groenendijk, T. Janssen, and M. Stokhof, eds. *Formal methods in the* *study of language* 277–322. Amsterdam: University of Amsterdam.

Kamp, Hans, and Reyle, Uwe (1993) *From discourse to logic: introduction to modeltheoretic semantics of natural language, formal logic and discourse representation theory.* Dordrecht: Kluwer.

Parsons, Terence (1990) *Events in the semantics of English.* Cambridge, Mass.: MIT Press.

Ramsey, Frank (1927) Facts and propositions. Reprinted in Ramsey, Frank, *Foundations: essays in philosophy, logic, mathematics, and economics* 40–57. Atlantic Highlands, N.J.: Humanities Press, 1978.

Reichenbach, Hans (1947) *Elements of symbolic logic.* New York: Macmillan.

Skolem, Thoralf (1928) Über die mathematische Logik. *Norsk matematisk tidskrift* 10:125–142.

Vendler, Zeno (1957) Verbs and times. In Vendler, Zeno, *Linguistics in philosophy* 97–121. Ithaca: Cornell University Press, 1967.