The Representation of Noun Phrases in Logical Form

Mary Patricia Harper
Ph.D Dissertation

Brown University
Dept. of Computer Science
Providence, Rhode Island 02912

Technical Report No. CS-89-46
May, 1990

The Representation of Noun Phrases in Logical Form

by
Mary Patricia Harper
B. A., Kent State University, 1976
M. S., University of Massachusetts at Amherst, 1980
Sc. M., Brown University, 1986

Thesis
Submitted in partial fulfillment of the requirements for the
Degree of Doctor of Philosophy in the Department of Computer Science
at Brown University.

May, 1990

Abstract

Several researchers in artificial intelligence have recognized the usefulness of a two-stage model of sentence comprehension for building a computer model of language. In the first stage, an intermediate level of representation called logical form is derived. During the second stage, logical form is updated with additional information (e.g., quantifier scoping). We introduce three constraints we consider necessary to make this model of language computationally feasible:

1. Logical form should compactly represent ambiguity.
2. Logical form should be initially computable from syntax and local (sentence-level) semantics. In particular, logical form should not be dependent on pragmatics, which requires inference and, hence, internal representation.
3. Further processing of logical form should only disambiguate or further specify logical form. Logical form has a meaning. Any further processing must respect that meaning.

Within this framework, we have devised logical form representations for pronouns, singular definite noun phrases, and singular indefinite noun phrases. For example, we represent a pronoun as a function of all of the variables corresponding to operators that can bind the pronoun. This representation allows us to indicate a meaning for the pronoun without deciding on the antecedent for the pronoun. Later, when we can determine the antecedent for the pronoun, we replace the pronoun function with the the variable or function used to represent its antecedent. Like pronouns, definites are represented as functions. However, indefinites cannot initially be represented as a function in logical form. Initially, we represent an indefinite as an existentially quantified variable. Later when more information is available about the meaning of a noun phrase, the initial representation is limited to indicate the intended meaning of that noun phrase.

We demonstrate that these representations both model the appropriate linguistic behavior and satisfy our computational constraints. This work has been implemented and tested on a wide variety of examples.

Acknowledgements

The hardest part of writing a thesis is properly thanking all those who have helped directly or indirectly from its beginning. I would particularly like to thank my advisor, Eugene Charniak. His help, insight, and trust have all made this thesis possible. I would also like to thank him for encouraging me to become an independent researcher. I would like to thank the members of my committee, Tom Dean and Leora Morgenstern. Their encouragement and excellent advice served to enrich the final version of this thesis. I would also like to thank Peter Wegner for his support and encouragement.

I would like to thank my husband Paul and my daughter Sarah for their patience and understanding while I was in graduate school. Without them, this thesis would not have been possible. I would also like to thank my mom and dad for teaching me the value of learning.

I would like to thank all of my fellow students at Brown who have listened to my ideas and offered encouragement and advice. I would like to thank my fellow AI students each of whom has enriched the AI research environment at Brown. In particular, I would like to thank John Arnold, Ken Basye, Mark Boddy, Mike Gavin, Robert Goldman, Keiji Kanazawa, Paul Lewis, Bobbo McCartney, Maury Neiburg, Kate Sanders, Lynn Stein, and Felix Yen. Finally, I would like to thank all of those unsung staff members at Brown who have offered friendship and help over the years.

This work has been supported in part by the National Science Foundation under grants IST 8416034 and IST 8515005 and by the Office of Naval Research under grant N00014-79-C-0529.

Contents

List of Figures

Chapter 1

Logical Form

1.1 Introduction

The central goal of this thesis is to provide a computational model of English capable of deriving the meanings of a wide variety of sentences. To achieve this goal, we adopt the hypothesis that parsed sentences should be mapped into an intermediate level of representation called logical form. By using logical form, we can partially specify the meaning of a sentence based on syntactic and sentence-level information, without considering the effect of pragmatics and context on the meaning of the sentence. Logical form allows us to *grow* the meaning of a sentence incrementally as more information becomes available.

For logical form to be useful in a computational framework, it should have certain properties. It should compactly represent some underdetermined meaning of a sentence, underdetermined in that ambiguities and anaphora are not resolved. Logical form should be derived using only syntactic and sentence-level information. Finally, when logical form is updated to indicate a single final meaning of a sentence, that update should be consistent with the initial logical form. The architecture for this approach to natural language comprehension is shown in Figure 1.1. Notice that three levels of representation are needed in this architecture, as well as three algorithms to map one representation into the next. In this thesis, we are primarily interested in specifying logical form. We examine how to map a parse tree into logical form. We also consider how logical form should be mapped into a final unambiguous internal representation.

Our departure in this thesis is not in our use of logical form. Many researchers in Linguistics and Artificial Intelligence have used an intermediate level of representation in natural language (e.g., Schubert and Pelletier [46], Allen [1], Sag [44]). However, we propose constraints on how logical form should be used in a computational model. These constraints shape our design of a sentence's logical form. They also indicate how logical form should be used in a computational model. In this thesis, we focus on the representation of singular noun phrases in logical form. The logical forms for noun phrases must obey our logical form constraints. They must also be able to model the range of behaviors those noun phrases can have. In fact, the noun phrase representations we develop provide us with a consistent way to handle a variety of examples that other approaches have difficulty with.

In the remainder of this chapter, we define more precisely what logical form is, discuss why it is useful in a computational model, and discuss why it was introduced in the first place. We also

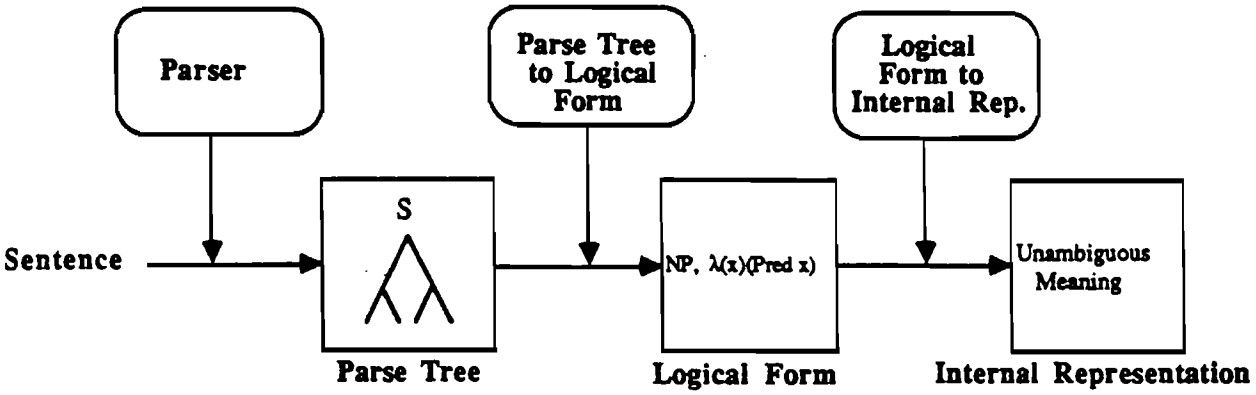

Figure 1.1: Language Architecture

summarize the contributions of this thesis, and discuss the organization of the remaining chapters.

1.2 Logical Form

Logical form is an intermediate level of representation between parse trees and internal representation[1]. Logical form has been quite popular within the Artificial Intelligence community (e.g., Schubert and Pelletier [46], Allen [1]) because it solves a serious computational problem. More semantic information can be gathered from a sentence than can be specified in a parse tree, but not enough is available to give an unambiguous internal representation for the sentence's meaning. For example, when logical form is derived from a sentence, each noun phrase is assigned a logical role (e.g., agent, patient, etc.) in a predicate-argument structure. In contrast, quantifier scoping and the antecedents of pronouns cannot be specified using only sentence-level information. Logical form provides a needed intermediate level between parse trees and internal representation. It allows us to represent a sentence before determining how other sentences affect its meaning, thereby separating disambiguation from the operation of the parser-translator. With further processing, logical form can be modified into a single unambiguous internal representation for the sentence's meaning.

One possible way to characterize the internal representation for a sentence would be to provide a single model-theoretic interpretation for it. To build a model theory we must be able to assign values to terms and formula in our language that correspond to things in the world. Not all representations of a sentence can be immediately assigned a single model-theoretic interpretation (without disjunction). For example, parse trees initially contain so little information (e.g., no antecedents for pronouns, etc.) that it is difficult to see how they would be assigned single non-disjunctive model-theoretic interpretations. Even though a parse tree encodes the structural properties of a sentence, much information must be determined before a single final meaning is available. Logical form is an intermediate level of representation, sharing many properties with a parse tree. Though

[1]Logical form has also been used to describe the representation of the unambiguous meaning of a sentence in logic (e.g., Nash-Webber and Reiter [37]), as well as a level of representation in government binding theory (see Chomsky [11]), (which combines features of parse trees with features of logic, with no indication of how to map into a final interpretation of a sentence). This is not the way we use the term logical form in this thesis.

some semantic information is specified in logical form, initially the meaning of the sentence is not fully developed. Hence, we would not expect the initial logical form of a sentence to provide a single model-theoretic interpretation for that sentence. However, logical form should be consistent with any of the possible model-theoretic interpretations of a sentence. Logical form is a schema, consistent with many possible meanings for a sentence.

1.2.1 Computational Benefits of Logical Form

By using logical form, the process of deriving the meaning of a sentence can be divided into several modules, as shown in Figure 1.1. A parse tree indicates the structural properties of a sentence. Logical form, on the other hand, provides information about the logical roles of noun phrases while preserving some syntactic information (e.g., the surface subject is identified). Once logical form is combined with contextual information, a single internal representation for the sentence's meaning can be provided.

To operationalize the model in Figure 1.1, we must specify each level of sentence representation. We must also specify how to map from one representation into the other. The parser is the first mapper in Figure 1.1. It takes a sentence as input and outputs a parse tree. In this thesis, we assume that the parser is already well specified. The second mapper uses information available in the parse tree (in combination with what we know about sentence-level semantics) to determine a sentence's logical form. Before indicating how this mapper works, we must determine how a sentence and its components should be represented in logical form. Additionally, we must indicate how logical form can be derived from the parse tree. Finally, the third mapper takes the logical form as input, and using contextual information outputs an unambiguous internal representation for the sentence. We examine some pieces of this mapper. However, because the effect of context on language comprehension is not fully understood, mapping logical form to internal representation is not completely specified in this thesis.

If the assumption that we can map into logical form independently of context is tenable, then mapping a parse tree to logical form should be much simpler than mapping from logical form into internal representation. The logical form for a sentence should be derived directly from the parse tree for that sentence. Logical form should maintain semantic ambiguity, allowing us to postpone decisions about antecedents of anaphoric noun phrases or quantifier scoping. This seems necessary, since determining the correct quantifier scoping for a sentence may require information available in subsequent sentences. Though antecedent candidates may be easy to locate, additional information is often necessary to pick a unique antecedent.

We propose three constraints on the use of logical form in a computational model of language comprehension:

1. Logical form should compactly represent ambiguity.
2. Logical form should be initially computable from syntax and local (sentence-level) semantics. In particular, logical form should not depend on pragmatics, which requires inference and hence internal representation.
3. Further processing of logical form should only disambiguate or further specify logical form. Logical form has a meaning. Any further processing must respect that meaning.

The first constraint (or the compactness constraint) expresses the spirit of using logical form in

a computational model of language comprehension. Logical form should not commit to a single meaning when there is semantic ambiguity in a sentence. Yet logical form should capture semantic ambiguity (e.g., quantifier scoping or pronoun reference) without resorting to explicit disjunctions of readings. In other words, constraint one indicates that redundancy should be minimized when we store the representation for a highly ambiguous sentence. The second constraint (or the modularity constraint) influences the mapping of parse trees into logical form. This constraint indicates that logical form should be generated independently of context. If contextual information is necessary to provide the initial logical form for a sentence, then logical form becomes superfluous. Why not use that information to directly generate an unambiguous meaning for the sentence? Finally, the third constraint (or the formal consistency constraint) affects the mapping of logical form to internal representation. When we provide the internal representation for a sentence, we should simply constrain the logical form for the sentence. These three constraints shape what logical form is, in addition to specifying how it should be derived and updated.

One of the major advantages of logical form is that it allows us to postpone decisions about the final meanings of ambiguous sentences. Several researchers have used logical form to handle quantifier scope ambiguity. Schubert and Pelletier [46] maintain quantifier scope ambiguities in logical form by placing quantified terms in the predicate-argument structure for the sentence, with the intention that quantifiers should be extracted and ordered in some later phase of sentence comprehension. Allen [1] employs a similar mechanism for maintaining quantifier scope ambiguities, though his method of specifying scoping information is a little different (as we will show shortly). Consider example 1.

Example 1

```
Someone loves everyone.

Meanings:
1. ∃x ∀y (love x  y)
2. ∀y ∃x (love x  y)
```

Despite the fact that this sentence has two meanings, both researchers provide a single logical form for this sentence, shown in 2.

Example 2

```
Someone loves everyone.

(love [∃x x] [∀y y])
Meaning:
(or ∃x ∀y (love x  y)
    ∀y ∃x (love x  y))
```

Because the quantifiers are stored in the predicate argument-structure with no scoping information indicated, this representation does not commit to one of the meanings indicated in 1. In fact, the logical form in 2 is simply a compact way of expressing the disjunction of readings in 1. Later, when more information is available, the initial logical form for the sentence is modified to indicate quantifier scope information.

Allen's method of indicating quantifier scoping differs from Schubert and Pelletier's method. For

example, assume *someone* has scope over *everyone* in example 1. Schubert and Pelletier [46] indicate this information by extracting the quantifiers to the left of the predicate-argument structure for the sentence, as shown below:

Example 3

Someone loves everyone.

$\exists x\ \forall y$ (love x y)

Allen [1] uses a much more expressive method for describing the quantifier scope information. Consider the representation in 4.

Example 4

Someone loves everyone.

(love [$\exists$x x] [$\forall y_x$ y])
Meaning:
$\exists x\ \forall y$ (love x y)

Allen indicates that the existential has scope over the universal by adding the existential's variable to the list of quantifiers (indicated as a subscript on the universal quantifier) that have scope over the universal. Hence, the formula in 4 expresses the same information as the formula in 3. However, Allen's method is not limited to a linear sequence of operators when expressing quantifier scoping information[2]. This issue concerns mapping to internal representation, however, not the design of logical form. Both approaches agree that quantifier scoping should not be initially specified in logical form, and both use the same mechanism to achieve this goal. Both approaches provide a single logical form for a sentence with quantifier scope ambiguity, yet the representation does not commit to a single meaning of that sentence. Hence, both researchers provide logical forms consistent with our compactness constraint. Their representations are also consistent with the modularity constraint. Each approach uses only syntax and knowledge about predicate-argument structures to derive the logical form for a sentence. Once the quantifier scoping is ascertained, each provides a way to modify logical form to indicate the intended meaning of the sentence. Because the modification of logical form simply commits to one of the meanings encoded in the logical form, each approach obeys the formal consistency constraint.

Another source of semantic underspecification in a sentence is anaphora. Antecedents for pronouns cannot be determined using only syntactic information. Contextual and syntactic information combine in a way that allows a language comprehender to determine a pronoun's antecedent. However, because syntax constrains the set of possible intrasentential antecedents, we should be

[2]Hintikka [21,20] has noted the fact that the linear ordering of quantifiers is not sufficient to capture all possible meanings of a sentence when four or more quantifiers occur in the sentence. Consider a sentence with four quantifiers, two universal and two existential.

Example 5
Every boy$_i$ wanted every girl$_j$ to introduce a friend of his$_i$ to a friend of hers$_j$.

It should be possible for *every boy* to have scope over *a friend of his*, without having scope over *a friend of hers*. Similarly, *every girl* could have scope over *a friend of hers*, but not *a friend of his*. There is no way to express this in a linear quantifier scoping string. $\forall$x: (boy x) $\exists$z: (friend-of x) $\forall$y: (girl y) $\exists$w: (friend-of y) won't work because $\forall$x has scope over $\exists$w. Similarly $\forall$y: (girl y) $\exists$w: (friend-of y) $\forall$x: (boy x) $\exists$z: (friend-of x) won't work since $\forall$y would have scope over $\exists$z. Allen has no trouble indicating nonlinear scoping using his approach.

able to provide logical forms for pronouns before their antecedents are known. In Chapter two, we provide a method to construct logical forms for pronouns. We design this representation to be consistent with our three logical form constraints. To be consistent with the modularity constraint, the logical form for a pronoun cannot commit to a specific antecedent for that pronoun. On the other hand, to be consistent with the compactness constraint, the logical form cannot consist of all of the possible antecedents. Finally, to obey the formal consistency constraint, the initial representation of a pronoun must be consistent with all the ways that pronoun can act. That way, when we update logical form to indicate the pronoun's antecedent, we simply constrain the logical form. Our representation of pronouns in logical form is a major contribution of this work. The development of this representation allows us to divide the process of determining a pronoun's antecedent into two phases. The first phase, handled by the syntax-to-logical-form mapper, indicates syntactic constraints on a pronoun. The second phase, handled by the logical-form-to-internal-representation mapper, determines what the antecedent is.

In summary, the usefulness of logical form seems obvious in a computational model of language comprehension. By using logical form, we divide the problem of sentence comprehension into three phases. Each phase is examined independently of the others, allowing us to tackle more manageable pieces of language comprehension. The three constraints on logical form are consistent with our goal of a computational model of language comprehension. The compactness constraint expresses space concerns. The other two concern the plausibility of computing logical form and incrementally updating it in a meaningful way. These constraints commit us to designing logical form in a way consistent with a computational model of language comprehension.

In the next section, we discuss some research conducted in the linguistics community concerning whether an intermediate level of representation is necessary to generate the meaning of a sentence. Though the necessity of logical form would add to our position in this thesis, we would not be disturbed if logical form was dispensable. We claim that because logical form is a useful construct for building a computer model of language comprehension, we will use it whether it is dispensable or not. We review this work to introduce this issue and to summarize work conducted in verb phrase ellipsis. We also introduce this work since it has influenced our representation of pronouns and definites.

1.2.2 Logical Form, Verb Phrase Ellipsis, and Pronouns

The question of whether logical form is necessary to describe legal mappings from syntax to final interpretation is an interesting one. On the one hand, there seems to be a computational motive for dividing the problem of language comprehension into the phases suggested in Figure 1.1. Logical form allows us to divide the process of language comprehension into incremental stages. When we map a sentence into logical form, we map the sentence into a formalism that can be updated based on what we learn in future sentences, what we know about the world based on previous sentences, and what we know in general. This division of labor allows us to indicate constraints on a sentence's meaning provided by its structure without constructing a list of possible meanings. However, if there is a way to directly specify the legal meanings of a sentence, then one might argue that the level of logical form is dispensable.

Verb phrase ellipsis is a linguistic phenomenon which some have argued requires an intermediate level of representation (e.g., Sag [44], Williams [51]). Verb phrase ellipsis is the deletion of a verb phrase in a sentence. The second sentence in example 6 is a sentence with verb phrase ellipsis (also called an elided sentence).

Example 6

Trigger Sentence: Fred$_i$ loves his$_i$ wife.
Elided Sentence: George$_j$ does too.
Meanings:
1. George loves Fred's wife.
2. George loves George's wife.

An elided sentence has little meaning independent of the first sentence (often called a trigger sentence). In this example, the index on *Fred* and *his* indicates that they are coreferential. Given that *his* refers to the subject of the trigger sentence[3] and the elided sentence depends on the trigger sentence for its meaning, the meaning of the elided sentence is ambiguous. It can either mean that *George loves Fred's wife* (known traditionally as the strict reading), or that *George loves his own wife* (known traditionally as the sloppy reading). The trigger sentence in example 6 constrains the available meanings of the elided sentence. If *his* refers to *Fred* in the trigger sentence, the elided sentence cannot mean that *George loves some other person's wife* (other than Fred's or George's). In verb phrase ellipsis, the meaning of the trigger verb phrase constrains the meaning of the elided verb phrase. The meaning chosen for a pronoun in the trigger verb phrase affects the meaning of the pronoun in the elided sentence.

In the next two sections, we describe two accounts of the linguistic phenomenon of verb phrase ellipsis. One approach relies on an intermediate level of representation to model the behavior of sentences with verb phrase ellipsis (i.e., Sag [44]). The other approach attempts to directly provide a model-theoretic interpretation for such a sentence (i.e., Partee and Bach [38]). The first approach provides a better model of verb phrase ellipsis than the second, adding support to the necessity of logical form. However, the problems of the second approach could result because of their pronoun representation, not because a level of logical form is indispensable.

Sag (1976)

Sag [44] examines conditions under which a verb phrase could be deleted in a transformational grammar model of language. He demonstrates that a deletion rule in syntax cannot account for all of the effects of verb phrase ellipsis. Sag instead proposes that any syntactic rule guiding verb phrase deletion should be constrained in some additional way. To suitably constrain syntax, Sag suggests that a verb phrase in a given sentence can be deleted (elided) only if the logical form of its verb phrase is equivalent in some way to the logical form of the verb phrase of the trigger sentence[4].

Before continuing with Sag's account of verb phrase deletion, we should point out that our use of the term logical form is a little different from Sag's use of the term. Like us, he uses logical form to define a level of representation between syntax and final interpretation. Unlike us, he assumes that pronouns are initially co-indexed with their antecedents, hence a large number of logical forms should be available for any sentence containing a pronoun. Because of this, his logical

[3]The antecedent of *his* could be *Fred* or some entity outside of the sentence. However, the index i on the noun phrases in example 6 allows us to concentrate on the meanings of the elided sentence that follow given that the pronoun refers to the syntactic subject of the trigger sentence. Other possible antecedents could be assigned to that pronoun, and our approach should also handle these readings of the elided sentence. The index does not mean that the antecedent of the pronoun *his* is syntactically assigned.

[4]Williams [51] develops a similar account for verb phrase ellipsis, but instead of devising a deletion rule for verb phrase ellipsis, he devises a rule to determine the meaning of elided verb phrases.

form violates several of the logical form constraints we suggest. In particular, his approach violates compactness and modularity. Determining antecedents of pronouns can be a complex problem requiring inference on the part of the language comprehender. We return to this issue at the end of this section.

The best evidence that verb phrase deletion must be constrained by some level of semantic identity (either at the level of logical form or at the level of meaning) is example 6. To derive the second reading of the elided sentence requires some mechanism which relates arguments in the sentence to each other. This relation cannot be captured by a syntactic rule. Sag [44] introduces additional evidence that verb phrase deletion requires logical form equivalence of verb phrases. Consider the sentence in example 7.

Example 7

```
Someone kissed everyone.
```

This sentence has two possible meanings. Because of the quantifier scope ambiguity, the sentence in 7 can be represented in either of the ways shown in 8.

Example 8

```
Someone kissed everyone.

a.  ∃y ∀x (kiss x y)
    ; One particular person kissed everyone.
b.  ∀x ∃y (kiss x y)
    ; Everyone was kissed by somebody (not necessarily the same one).
```

Now consider the effect of conjoining the sentence in example 7 with another sentence of similar form.

Example 9

```
Someone kissed everyone, and Sarah kissed everyone.
```

The second conjunct of example 9 is not ambiguous since the universal does not affect the meaning of *Sarah*. However, the first conjunct of the example is still ambiguous. Even though we prefer the reading where the universal gets narrowest scope in the first conjunct, we can still get the wide scope reading. However, once the verb phrase of the second conjunct is elided (shown in 10), the first conjunct is no longer ambiguous.

Example 10

```
Someone kissed everyone, and Sarah did too.
```

In example 10, the existential must have scope over the universal in the trigger clause. Based on a syntactic theory alone, one would not expect verb phrase ellipsis to limit the number of meanings of the trigger sentence. Quantifiers in the two conjuncts should be able to act independently. This example can be explained given Sag's thesis that verb phrase ellipsis is conditioned on the equivalence of logical form. Before showing this, we discuss Sag's logical form.

The logical form of a sentence is a predicate-argument structure. Sag's logical form maintains

some surface information (e.g., subject-predicate information). It also indicates some of the logical properties of a sentence (e.g., logical roles of noun phrases). The sentence's subject is distinguished from other noun phrases in the sentence. To distinguish the subject of a sentence, Sag [44] borrows from Church's lambda calculus [12]. The subject is lambda abstracted from the predicate-argument structure for the sentence. When the subject is abstracted from the predicate-argument structure, it is replaced by the variable corresponding to the lambda operator. Consider how Sag would represent the sentence in example 11.

Example 11

```
Mary loves Sarah.

Mary, λ(x)(love x Sarah)
```

The surface subject of the sentence is abstracted from the predicate-argument structure for the sentence. The role of the subject in the predicate-argument structure is indicated with the lambda variable, x. The noun phrases in this example are represented as strings. Sag's logical form for a sentence has a very nice property. The lambda function is the representation of the sentence's verb phrase. By applying the surface subject to the lambda function, Sag also provides the logical form for a sentence. Usually in lambda calculus, the terms are applied to the right of a lambda function, as shown below:

Example 12

```
Mary loves Sarah.

λ(x)(love x Sarah) Mary
```

However, Sag chose to apply terms on the left of the function because it is easier to see the correspondence between the representation of a sentence and that sentence. In the rest of this thesis, we will do the same.

To handle verb phrase ellipsis, Sag requires that all sentences are represented in logical form. He proposes that once the syntactic requirements for legal verb phrase ellipsis are met, the verb phrase of a sentence S can be deleted, if and only if, the lambda function corresponding to the verb phrase of S is an alphabetic variant of some other lambda function in S or in S', which precedes S in discourse. In other words, verb phrase ellipsis is possible only when the logical form for a verb phrase in the trigger sentence is equivalent in some manner to the logical form for the verb phrase which is a candidate for deletion. Consider Sag's definition of alphabetic variants.

> Intuitively, two λ-expressions are alphabetic variants if they differ only with regard to variable letters. The notion is not quite this simple, however. For two λ-expressions, λx(A) and λy(B), to be alphabetic variants, every occurrence of x in (A) must have a corresponding occurrence of y in (B), and vice versa. Also, any Quantifier in A that binds variables (in A) must have a corresponding (identical) Quantifier in B that binds variables in all the corresponding positions (in B). However, if there are any variables in A that are bound by some quantifier outside of λx(A), then the corresponding variable in λy(B) must be bound by the same operator in order for alphabetic variance to obtain (λx(...) and λy(...) are alphabetic variants in (∀z)[John, λx(x loves z) and Bill, λy(y loves z)]). Crucially, if λx(A) contains a variable bound outside of λx(A) (for instance, z in (∀z)[John, λx(x loves z)]) and λy(B) contains a corresponding variable bound outside

> of λy(B) (even one bound by an analogous operator, for instance w in (∀w) [John, λy(y loves w)]) the two λ-expressions are not alphabetic variants (though here the universally quantified expressions, considered as a whole, would be). (pp. 72-73)

Sag's model indicates that the only possible meanings for an elided sentence are those whose verb phrase representations are alphabetic variants of one of the trigger sentence's verb phrase representations. Sag's model predicts when certain meanings for an elided sentence are impossible. It also predicts when certain representations of a trigger sentence cannot sanction ellipsis. For instance, Sag uses alphabetic variant equivalence to account for the ambiguous scoping example discussed earlier (i.e. examples 7, 8, 9, and 10). The verb phrase of the sentence *Sarah kissed everyone* can be deleted only when the universal operator in the trigger sentence is under the scope of the lambda and existential operators, as shown in 13[5].

Example 13

```
∃x  x, λ(y)[∀z (kiss y z)]
Sarah, λ(m)[∀n (kiss m n)]
```

On the other hand, if the universal has scope over the existential, the lambda functions are no longer alphabetic variants, as shown in 14.

Example 14

```
∀z ∃x  x, λ(y)(kiss y z)
∀n Sarah, λ(m)(kiss m n)
```

Hence, Sag's model predicts that the meaning of the trigger sentence indicated in 14 cannot be available when it is used to sanction verb phrase ellipsis in *Sarah kissed everyone*. Sag's model of verb phrase ellipsis explains many other interesting linguistic phenomena, including comparative ellipsis, ellipsis given WH-questions, and sloppy identity ambiguity.

Sag's model accounts for sloppy identity ambiguity in a very nice way. The meaning of an elided sentence is ambiguous when its trigger sentence contains pronouns that refer to the subject of that sentence. Example 6 is a good example of the sloppy identity ambiguity.

Example 6

```
Trigger Sentence: Fred_i loves his_i wife.
Elided Sentence:  George_j does too.
Meanings:
1. George loves Fred's wife.
2. George loves George's wife.
```

Given that *his* refers to the subject *Fred*, the elided sentence is ambiguous. Sag claims that the ambiguity of the elided sentence in example 6 results because the trigger sentence, though not ambiguous, can be represented in two equivalent, but different ways. The trigger sentence can be represented in two ways because the pronoun whose antecedent is the subject can be represented in two different ways. Either the pronoun is co-indexed with the subject (see 15a) or it is replaced by the lambda variable corresponding to the subject (see 15b).

[5]For alphabetic variance to hold in this example, the universal corresponding to *everyone* must remain inside the verb phrase representation.

Example 15

Trigger Sentence: Fred$_i$ loves his$_i$ wife.

a. **Fred$_i$, λ(x)(love x his$_i$ wife)**
   ; Fred loves Fred's wife (*his* co-indexed with *Fred*).
b. **Fred$_i$, λ(x)(love x x's wife)**
   ; Fred loves Fred's wife (*his* replaced by bound variable).

The ellipsis of the verb phrase in the sentence *George loves his wife* is allowed only when the logical form for its verb phrase is an alphabetic variant of the verb phrases in 15a or 15b. The sentence *George loves his wife* can be represented in a variety of ways. We do not necessarily know the antecedent for the pronoun *his*. Hence, four possible representations for the sentence are possible, as shown in 16.

Example 16

Candidate Sentence: George loves his wife.

a. **George$_j$, λ(y)(love y his$_i$ wife)**
   ; George loves Fred's wife
b. **George$_j$, λ(y)(love y y's wife)**
   ; George loves George's wife
c. **George$_j$, λ(y)(love y his$_j$ wife)**
   ; George loves George's wife
d. **George$_j$, λ(y)(love y his$_k$ wife)**
   ; George loves a third party's wife

Since the lambda functions in 15a and 16a are alphabetic variants, the verb phrase in *George loves his wife* can be deleted. Additionally, one possible meaning for the elided sentence is *George loves Fred's wife*. Likewise, because the lambda functions in 15b and 16b are alphabetic variants, again ellipsis is licensed. Hence, another possible meaning for the elided sentence is *George loves George's wife*. On the other hand, since the trigger verb phrases in 15 are not alphabetic variants of the lambda functions in 16c[6] or 16d, the elided sentence cannot receive either of these meanings (given the fact that *Fred* is the antecedent for *his* in the trigger sentence). Thus Sag's model provides an explanation for the sloppy identity ambiguity.

Sag's model also handles example 17. This example highlights the necessity of maintaining surface structure information in the logical form of a sentence.

Example 17

Trigger Sentence: Fred was hit by the apple.
Candidate Sentence: The orange hit Fred.

The trigger sentence indicated in example 17 cannot sanction deletion of the verb phrase of the

[6]Though the apparent meaning of the reading in 16c is the same as the reading in 16b, the reading in 16c is not sanctioned. One reasonable explanation for this is that the meaning of the elided sentence might be used to generate the meaning of a subsequent elided sentence. For example if we say, *Fred loves his wife. George does too. Harry does too*, and we sanction the reading *George loves George's wife* then we should also sanction the reading *Harry loves Harry's wife*, and we should not sanction *Harry loves George's wife.*

candidate sentence. To see that this is the case, notice the oddness of the following example.

Example 18

Trigger Sentence: Fred was hit by the apple.
Elided Sentence: *The orange did too.

These two sentences are incompatible. This incompatibility is easily discovered by examining Sag's logical forms for the trigger and candidate sentences from example 17, shown below:

Example 19

Trigger Sentence: Fred was hit by the apple.
Fred, λ(x)(hit *the apple* x)
Candidate Sentence: The orange hit Fred.
***the orange*, λ(x)(hit x Fred)**

Notice that the predicate-argument structure for the trigger sentence is different than the predicate-argument structure for the candidate sentence. Because the verb phrases are not alphabetic variants of one another, Sag's model correctly rules out the possibility of deletion in example 17.

As we pointed out earlier, Sag's logical form differs from ours. In Sag's approach, some noun phrases receive logical representations, others do not. Additionally, Sag's logical form contains information concerning antecedents of pronouns. Contextual information is often needed to determine this information. Williams' model of verb phrase ellipsis [51] is more incremental than Sag's. Williams divides processing of logical form into two phases. In particular, he specifies that all Sentence Grammar Rules must apply before Discourse Grammar Rules. Sentence Grammar Rules are used to determine the meaning of a sentence (e.g., antecedents for pronouns or quantifier scoping). On the other hand, Discourse Grammar Rules include a rule to recover the meaning of a null verb phrase. William's null verb phrase rule requires that the antecedent verb phrase and the null verb phrase share certain characteristics (like alphabetic variant equivalence). This division seems to capture the necessary steps required to determine the meaning of a sentence with verb phrase ellipsis. First, the meaning of the trigger must be specified, including some indication of the antecedents of the pronouns in the sentence. Once this meaning is available, then it should be possible to determine the meaning of the elided sentence.

Partee and Bach (1981)

Partee and Bach [38] attempt to dispense with logical form in translating from syntax to final interpretation. They define a method to provide model-theoretic interpretations for their initial representation of pronouns and elided verb phrases. In their approach, null verb phrases and pronouns are initially represented as variables. Later, these variables are assigned values depending on their type. However, difficulties arise in this approach because of an interesting interaction between the assignment of values for pronouns and the assignment of values for the null verb phrase.

For the reader to understand this interaction, we briefly review Partee and Bach's approach. Partee and Bach [38] assume Montague's [35] general theory (with a few modifications to get around the strict compositionality of that approach). All of the possible representations for ambiguous sentences are simultaneously generated, avoiding the need for an intermediate level of representation.

For example, if a sentence has two possible quantifier scopings, two meanings are generated for the sentence.

Partee and Bach represent pronouns as variables. These variables are either bound by some operator or remain free within the representation of the sentence. If a pronoun is not bound, it is assigned some value by a context assignment function, that is, a function which maps the variable to the individual that the pronoun denotes. Consider the representation of the two sentences in 20.

Example 20

a. John loves him.
   (love x)(John)
b. Every man loves himself.
   ∀y (if (man y) (love y)(y))

Notice that the subject is separated from (and placed to the right of) the verb phrase in the representation of each sentence. Partee and Bach's parser processes verb phrases independently of subjects to maintain compositionality. Later, the representations of the subject and verb phrase are combined to provide the meaning of the sentence. In 20a, the pronoun is represented as the variable x. Notice that x is not bound by an operator. Because x is unbound, Partee and Bach rely on a context assignment function to provide the variable with a value. For instance, the context assignment function could assign x the value *Fred Jones*. In contrast, in 20b, the pronoun is represented as the variable y, which is bound by a universal quantifier. Because y is bound, the context assignment function is unnecessary.

Partee and Bach represent an elided (or null) verb phrase as a free property variable. A free property variable is typed to receive a value corresponding to a verb phrase already in discourse. It receives its interpretation in much the same way as an unbound pronoun variable, with the exception that the antecedent for the null verb phrase must be available in linguistic context. Once the value of the null verb phrase is provided, the meaning of the elided sentence is determined by combining the elided sentence's subject with the null verb phrase's assigned meaning.

We demonstrate how Partee and Bach's approach provides the meanings for elided sentences with an example. We simplify their representations to make our point without discussing unimportant details. Consider how the following example would be handled:

Example 21

Trigger sentence: Fred thinks he is sick.
Elided sentence: Bill does too.

Partee and Bach would represent the first sentence in 21 in (at least) two ways, as shown in 22.

Example 22

Fred thinks he is sick.

a. Fred, λx[(think [(sick) (x)])(x)]
b. Fred, λx[(think [(sick) (y)])(x)]
   or equivalently
   (think [(sick) (y)])(Fred)

In 22a, the pronoun variable is bound by the lambda operator. In 22b the pronoun is a free variable. Both of these representations are generated directly by the parser. The meaning of the representation in 22a is *Fred thinks that Fred is sick.* On the other hand, the meaning of the representation in 22b depends on the context assignment function. It could mean *Fred thinks that Fred is sick* or it could mean *Fred thinks that* **some other value assigned by the context assignment function** *is sick.* Partee and Bach use lambda abstraction of syntactic subjects as an optional rule. This rule can be applied to provide one of the possible meanings of a pronoun in a sentence. If the lambda operator does not bind a pronoun variable, the use of the lambda operator is unnecessary (as shown in 22b).

Assuming that some function provides meanings for null verb phrases, Partee and Bach's model can provide a meaning for the null verb phrase of the elided sentence in example 21. Because the function provides a meaning for the null verb phrase from linguistic context, it must locate the antecedent verb phrase. Assuming that either of the verb phrase representations in 22 can be the value for the null verb phrase, the elided sentence can receive either of the following representations.

Example 23

```
Bill does too.

a. Bill, λx[(think [(sick) (x)])(x)]
b. Bill, λx[(think [(sick) (y)])(x)]
            or equivalently
   (think [(sick) (y)]) (Bill)
```

The representation in 23a provides the sloppy interpretation of the elided sentence. Since the pronoun is lambda bound, this is the only possible interpretation. The other representation for the elided sentence, shown in 23b is a little more interesting. In this case, there is an unbound pronoun variable. In fact, in the representations of both the trigger (i.e. 22b) and elided (i.e. 23b) sentences, the pronoun is represented as an unbound variable which must be assigned a value by the context assignment function. This variable could be assigned any individual in the model, including Fred, Bill, or someone else. Hence a variety of interpretations for the elided sentence can be derived from this representation, depending on the context assignment function. Partee and Bach require that the context assignment function assign a free pronoun variable in meaning of a null verb phrase the same value as in the trigger verb phrase. Hence, the assignment function must assign the same value to the unbound variable y in 22b and 23b.

A level of logical form seems unnecessary in this approach because meanings are generated directly from the sentence. The context assignment function is used to assign values to null verb phrases and to unbound pronoun variables. However, Partee and Bach's approach does not work very well for all examples. Partee and Bach discuss three examples for which their approach fails. In each example, it is possible to assign an incorrect meaning to the elided sentence because of the way that pronoun variables are assigned their values.

Consider the first example.

Example 24

```
Trigger Sentence: No man believes that Mary loves him.
Elided sentence:  But she does.
```

This example presents a problem for Partee and Bach when *him* refers to *no man* in the trigger sentence[7]. Consider this meaning (shown in 25).

Example 25

```
No man believes that Mary loves him.

∀x (if (man x) ¬[(believe [(love x) (Mary)]) (x)])
```

Because the trigger sentence can be represented as shown above, the assignment function can assign the null verb phrase of the elided sentence in 24 the value (love x). Hence, one possible representation for the elided sentence is shown in 26.

Example 26

```
But she does.

(love x)(z)
; Mary loves Joe Johnson. (one possibility)
```

Notice that the context assignment function must assign values to the unbound pronoun variables, z and x. Hence, one possible interpretation of the elided sentence is *But Mary loves Joe Johnson*. However, no reading for the elided sentence should be available when *no man* is the antecedent for *him* in the trigger sentence. Yet, Partee and Bach's approach provides one.

The next example is slightly different. In this case, the representation of the elided sentence contains a universal operator which can accidentally bind a free pronoun variable in the value assigned to its null verb phrase.

Example 27

```
Trigger Sentence: Mary loves him.
Elided sentence:  Everyone assumes that Sally does.
```

Consider the meaning of the trigger sentence, shown in 28.

Example 28

```
Mary loves him.

(love z) (Mary)
```

In this case, no operator locally binds the pronoun variable. Assuming that *everyone* in the elided sentence is represented using a universal operator on z and the context assignment function assigns (love z) as the value of the null verb phrase, the meaning provided for the elided sentence is shown in 29.

[7] We do not star the elided sentence in this example since *him* might refer to someone outside of the sentence. In such a case, the example is fine.

Example 29

Everyone assumes that Sally does.

∀z (if (person z) [(assume [(love z)(Sally)]) (z)])
; Everyone$_i$ assumes that Sally loves him$_i$.

This meaning is clearly incorrect. However, Partee and Bach's approach provides it as a possible meaning of the elided sentence. To avoid readings like this, they must provide some way to ensure that operators contained in the meaning of the elided sentence do not bind variables in the meaning assigned to the null verb phrase.

Finally, in the case of conjoined sentences, there is nothing to rule out the same sort of odd reading as in example 27.

Example 30

Bill believes that Sally will marry him, but everyone knows she won't.

In particular, the trigger sentence could be represented as shown in 31.

Example 31

Bill believes that Sally will marry him.

Bill, λw[(believe [(marry w)(Sally)]) (w)]

Given that *everyone* is represented using a universal over w and the assignment function assigns (marry w) as the value of the null verb phrase in 30, the elided sentence is represented as shown in 32.

Example 32

But everyone knows she won't.

∀w (if (person w) [(know [¬[(marry w) (z)]]) (w)])
; Everyone$_i$ knows she won't marry him$_i$.

This reading is not a reasonable meaning for the elided sentence.

Because of these three examples, Partee and Bach concluded that logical form may be indispensable in a model of language comprehension[8]. Hence, the level of logical form seems necessary to account for verb phrase ellipsis. In any case, the usefulness of logical form in a computational model is clear. We do not wish to have a collection of all possible meanings for a sentence. These meanings would have to be filtered to find the correct meaning. This division of labor is not computationally feasible.

1.3 Contributions

In this section, we briefly summarize the contributions of this thesis.

[8]Despite the negative evidence, their framework could always be fixed. Their failure may simply indicate that their representation of pronouns is inadequate for modeling the ways that pronouns behave in English.

In this chapter, we have defined constraints on logical form for a computational model. These constraints are used extensively throughout this thesis to decide exactly how to represent noun phrases in logical form, as well as how to further specify logical form. These constraints both express the philosophy of our use of logical form and determine the nature of the logical form that we devise. As such, they are a central idea of this thesis.

A second contribution of this thesis is our representation of pronouns as functions in logical form. This representation requires that we know in advance (based on syntax and the mapping of syntax into logical form) when a pronoun in a sentence can act as a variable bound by some operator. We discuss previous work in linguistics suggesting that only syntactic information is required to determine when a pronoun can be bound by a noun phrase in the same sentence. Using this idea, we represent pronouns in a way consistent with our three constraints on logical form, with particular attention to the sloppy identity problem in verb phrase ellipsis. By using a function to represent pronouns, we are able to model the variety of behaviors pronouns have using a single representation.

In addition to representing pronouns in logical form, we provide logical form representations for definite and indefinite noun phrases. Each type of noun phrase is represented in a way that accounts for its linguistic behavior in English and in a way consistent with our constraints on logical form.

Another contribution of this thesis is the modification of the multiple pronoun constraint proposed in Sag [44]. This constraint limits the number of meanings an elided sentence can have when its trigger sentence contains two or more pronouns that refer to the same syntactic subject of a sentence. Finally, we describe the implementation of a system that we devised to provide logical forms for a variety of examples and to derive the meanings of elided sentences given their trigger sentences. This implementation demonstrates the usefulness of the noun phrase representations we devised. It also demonstrates that the logical form for a sentence can be generated using only syntactic and sentence-level information.

1.4 Organization of Thesis

This dissertation is divided into six remaining chapters. Chapters two, three, and four discuss the representation of pronouns, definite noun phrases, and indefinite noun phrases, respectively. These three chapters are organized similarly. First, we discuss the behaviors of the noun phrase we wish to model. Next, we present our representation of that noun phrase, discussing how our three constraints and linguistic evidence motivate our choice. Finally, we compare our representation with those suggested by other researchers. We show how our representation handles examples that the other approaches fail to cover.

In Chapter five, we discuss our modification of Sag's multiple pronoun constraint. In Chapter six, we discuss the implementation of a system which determines the logical form for sentences as well as the meanings of elided sentences. Finally, in the concluding chapter, we discuss our contributions and suggest future research directions.

Chapter 2

Pronouns

2.1 Introduction

In this chapter, we discuss the representation of pronouns in logical form. We motivate this representation using the constraints on logical form proposed in Chapter one, as well as linguistic evidence on the nature of pronouns. First, we discuss the behaviors of pronouns in English, then we present our representation of pronouns (taking into account pronoun behavior as well as our constraints on logical form, proposed in Chapter one). Finally, we compare our approach with previous attempts to model pronoun behavior.

2.2 Pronouns: Linguistic Evidence

2.2.1 Traditional Pronoun Dichotomies

Pronouns are always anaphoric. Either they have a linguistic antecedent or they depend on some salient individual in the environment of the speaker or hearer (deictic use of pronouns). Pronouns with linguistic antecedents can be categorized in two ways. Either their antecedents occur in the same sentence (called *intrasentential reference*[1]) or in other sentences (called *intersentential reference*). Pronouns with intersentential antecedents seem to act differently than pronouns with intrasentential antecedents, as we will show.

When the antecedent of a pronoun is represented as a universally quantified variable and it occurs in the same sentence as the pronoun, then the pronoun acts like a universally quantified variable (quantified over by the same operator as its antecedent). Consider the following example:

Example 33

Fred$_i$ showed every girl$_j$ her$_j$ picture.

Given that the antecedent for *her* is *every girl*, the pronoun seems to act like a variable bound by the universal quantifier of its antecedent. Hence, the sentence should be represented as shown

[1]By reference, we do not mean that the pronoun denotes its linguistic antecedent. Instead, we mean it depends on the meaning of its antecedent for its meaning. In other words, a pronoun adopts the behavior of its antecedent.

in example 34.

Example 34

```
Fred_i showed every girl_j her_j picture.

∀x (if (girl x) (show Fred (x's picture) x))
```

Here the agent is the first argument in the predicate-argument structure, the patient the second, and the recipient the third.

In contrast, if a pronoun's antecedent occurs in another sentence, then that pronoun cannot act like a bound variable. Quantifiers cannot have scope across sentences in English. Consider example 35.

Example 35

```
Fred_i likes everyone_j.
*He_j likes him_i.
```

The pronoun *he* in the second sentence of example 35 cannot be bound by the quantifier introduced to represent *everyone* in the first sentence. This is indicated by placing a star next to the second sentence. However, a quantified noun phrase in one sentence can be the antecedent for a pronoun in another sentence. Consider example 36.

Example 36

```
Fred_i likes everyone.
They like him_i.
```

In 36, *everyone* is the antecedent for *they*. However, *they* does not act like a quantified variable. Instead, it seems to refer to the group of individuals that *everyone* quantifies over. Webber [49] discusses how to construct discourse entities[2] for noun phrases in a sentence (both quantified noun phrases and noun phrases that are quantified over). If a noun phrase is the antecedent for a pronoun in another sentence, the pronoun is replaced with the antecedent's discourse entity. In 36, the noun phrase *everyone* evokes a discourse entity denoting a group of individuals. Hence, a plural pronoun in a subsequent sentence can have that universal noun phrase as its antecedent. Given that the antecedent for *they* is *everyone*, the pronoun simply adopts the discourse entity created for *everyone* as its meaning.

Pronouns have also been classified as bound variable or referential pronouns. The antecedent for a bound variable pronoun occurs in the same sentence as the pronoun and the meaning of the pronoun is represented as the variable bound by the operator associated with its antecedent. The pronoun *her* in 33 is an example of an intrasentential bound variable pronoun. In contrast, the meaning of a referential pronoun should be represented by the discourse entity evoked by its antecedent. The pronoun *they* in 36 is an example of an intersentential referential pronoun.

The bound versus referential dichotomy divides the world of pronouns in a slightly different way than the intersentential-intrasentential dichotomy. Pronouns with intersentential antecedents are

[2] A discourse entity is created for noun phrases when a sentence has been disambiguated. If a noun phrase is not anaphoric, then a new discourse entity is created for the noun phrase. A discourse entity is a designator for the entity or set of entities the noun phrase evokes in the discourse model of the speaker or hearer.

typically referential[3]. However, pronouns with intrasentential antecedents cannot always be classified as bound variable pronouns. Some researchers assume that pronouns whose antecedents are proper nouns are referential pronouns (e.g., Sag [44], Webber [49]). Additionally, some pronouns with intrasentential antecedents cannot be classified as bound variable pronouns or referential pronouns. Consider example 38.

Example 38

Every man$_i$ gave the psychiatrist who cares for (his$_i$ mother)$_j$ her$_j$ diary.

Given that *every man* is the antecedent for *his*, *his mother* is not referential. Hence, given that the antecedent for *her* is *his mother*, the pronoun cannot be referential. On the other hand, if we represent *his mother* as a quantified term, we might be able to replace *her* with a variable. However, in English, the complex noun phrase constraint (introduced by Ross [42]) prevents anything from moving out of a relative clause attached to a noun phrase, including quantifiers. Hence, the quantifier introduced to represent *his mother* would not be able to bind the pronoun *her*. The complex noun phrase constraint holds for universal quantifiers, as the following example demonstrates:

Example 39

*Fred$_i$ gave the psychiatrist who cares for every woman$_j$ her$_j$ diary.

The antecedent for *her* cannot be *every woman* in this sentence[4]. Similarly, in example 38, we assume that *her* cannot be a bound variable pronoun because of the complex noun phrase constraint. Additionally, the pronoun cannot be referential since it does not act like a constant (given the coreferences indicated in 38). Therefore, example 38 suggests that pronouns behave in some additional way not captured by the bound-referential dichotomy[5].

Next, we consider how pronouns behave in verb phrase ellipsis. We introduce several examples that also suggest the bound-referential dichotomy should be questioned.

2.2.2 Pronouns in Verb Phrase Ellipsis

The representation of pronouns is especially important in the domain of verb phrase ellipsis. Consider the available readings of the elided sentence in example 40, given that the antecedent for *his* is the subject in the trigger sentence.

[3]The only exception are pronouns like those in *paycheck sentences* (first noticed by Kartunnen [30]). Consider:

Example 37
Fred gave his paycheck to his wife.
George gave it to his mistress.

The pronoun *it* is not referential (and for that matter, it is not bound).

[4]The star in front of the sentence indicates that the indices on *every woman* and *her* are impossible in English.

[5]Kartunnen [30] also questions this dichotomy.

Example 40

```
Trigger Sentence: The policeman_i loves his_i wife.
Elided Sentence:  The milkman_j does too.
Meanings:
1. The milkman loves the policeman's wife.
2. The milkman loves the milkman's wife.
```

The elided sentence in 40 can only have two meanings (given that *the policeman* is the antecedent for *his*). Though the meaning of the elided sentence is ambiguous, it cannot mean that *The milkman loves some other person's wife* (given the indices on the noun phrases in the trigger sentence). The trigger sentence limits the meanings of the elided sentence. This suggests that the meanings of an elided verb phrase should be derived from the possible meanings of its trigger verb phrase.

To derive the two meanings of the elided sentence from the trigger sentence, two different meanings for the pronoun *his* must be provided (given that the antecedent for *his* is the subject). If the syntactic subject in the logical form of a trigger sentence is lambda abstracted (following Sag [44], Williams [51], Webber [49], and Partee and Bach [38]), the pronoun can refer indirectly to the subject if it is replaced by the subject's lambda variable. In 41, the pronoun *his* is replaced with the variable x, bound by the lambda operator which abstracts the subject of the trigger sentence.

Example 41

```
The policeman_i loves his_i wife.

the policeman, λ(x)(love x (x's wife))
```

To provide the sloppy meaning of the elided sentence, the pronoun in the trigger verb phrase must refer indirectly to the subject. Hence, the representation of the verb phrase in 41 is used to provide the sloppy reading of the elided sentence of 40 (i.e., the second meaning listed in example 40). The sloppy meaning for the elided sentence is shown in example 42.

Example 42

```
The milkman does too.

the milkman, λ(x)(love x (x's wife))
; The milkman loves the milkman's wife.
```

To derive the strict meaning of the elided sentence (i.e., the first meaning in 40), the pronoun *his* must refer to the subject *the policeman* in another way. We could represent *the policeman* as a quantified term and replace the pronoun *his* with the quantified variable. However, this is not a good solution, as we will show. Consider representing *the policeman* as ∃!x: (policeman x)[6], meaning ∃x: [and (policeman x) (∀y (policeman y) ↔ (y = x))]. Given that the antecedent for *his* is *the policeman*, we can replace the pronoun with the variable x.

[6]The colon separating the quantifier from its restriction is syntactic sugar. As shown in 43, the colon expands to conjoin the restriction with the meaning of the sentence.

Example 43

```
The policeman_i loves his_i wife.

∃!x: (policeman x)  x, λ(y)(love y (x's wife))
Meaning:
∃!x (and (policeman x)  x, λ(y)(love y (x's wife)))
```

This representation of the trigger sentence is fine, but consider using the lambda function in 43 to provide a meaning for the elided sentence.

Example 44

```
The milkman does too.

∃!z: (milkman z)  z, λ(y)(love y (x's wife))
; contains unbound variable  z
```

This representation of the elided sentence contains an unbound variable. In other words, it is ill-formed. This is one reason that Webber [49] and Sag [44] treat pronouns whose antecedents are definite noun phrases as referential pronouns. If *his* is replaced with a discourse entity for *the policeman*, then the strict reading of the elided sentence can be provided. This representation of the trigger sentence is shown in 45.

Example 45

```
The policeman_i loves his_i wife.

the policeman, λ(x)(love x (police_22's wife))
```

By applying the subject of the elided sentence to the lambda function in 45, the strict meaning of the elided sentence, shown in 46, is provided.

Example 46

```
The milkman does too.

The milkman, λ(x)(love x (police_22's wife))
; The milkman loves the policeman's wife.
```

In verb phrase ellipsis, when the antecedent for a pronoun is the trigger sentence's subject, the pronoun can act in two different ways, either as a lambda variable or as something depending on the subject's type. Two meanings of the elided sentence in example 40 are provided if we represent the pronoun *his* as either a lambda variable or a discourse entity. However, not all definite noun phrases can be represented as or referred to as a constant. Consider example 47.

Example 47

```
Every man_i asked (his_i mother)_j about her_j problems.
```

Given that *every man* is the antecedent for *his*, *his mother* cannot be referential. And given

that *his mother* is the antecedent for *her*, the pronoun cannot be referential. Hence, the pronoun cannot be replaced with a discourse entity. We return to this issue in section 2.4, when we discuss the approaches of Sag [44], Webber [49], Reinhart [40], and Partee and Bach [38].

2.2.3 Constraints on Bound Variable Anaphora

In the next two sections, we review some linguistic research which specifies when a noun phrase can bind a pronoun. The first approach uses syntactic information found in a parse tree, and the second uses sentence-level information available to the parse-tree-to-logical-form mapper.

Syntactic Approach: Reinhart (1983)

Reinhart [40] attempts to specify precisely when a certain noun phrase can be the antecedent for a pronoun. She distinguishes coreference (corresponding to referential pronouns) from bound anaphora (corresponding to bound variable pronouns). A pronoun is a bound anaphora when its antecedent falls in the same sentence and the meaning of the pronoun is represented using the variable corresponding to its antecedent. In contrast, Reinhart claims that coreference can occur across or within sentences. Coreference occurs when two noun phrases are assigned the same referent. Reinhart claims that coreference is a discourse level phenomenon, requiring contextual information and pragmatics. Hence, coreference cannot be modeled by using a simple syntactic rule. However, Reinhart claims that bound anaphora can be handled at the level of syntax. She specifies a syntactic rule to determine whether a certain pronoun can be bound by a certain noun phrase. By distinguishing between bound anaphora and coreference, Reinhart is able to provide a partial syntactic solution for anaphora.

Reinhart specifies a syntactic rule for determining when a pronoun can be bound by a noun phrase in the same sentence. She introduces c-command (or constituent-command), which is a relation between nodes in a parse tree, to provide the rule.

> Node A c(constituent)-commands node B iff the branching node α_1 most immediately dominating A either dominates B or is immediately dominated by a node α_2 which dominates B, and α_2 is of the same category type as α_1. (p. 23)

For example, consider the parse tree in Figure 2.1. Notice that NP1 c-commands *his* but NP2 and NP3 do not c-command *his*. Reinhart claims that a pronoun can be bound by a noun phrase if only if the noun phrase c-commands the pronoun. Hence, she claims that *every man who saw every boy* can bind *his*, but *every boy* cannot bind *his*. She claims that definite noun phrases can also bind pronouns and proposes that all noun phrases can bind pronouns by virtue of a lambda operator. She claims that any noun phrase can potentially bind those pronouns that it c-commands.

Reinhart distinguishes between the behaviors of reflexive and non-reflexive pronouns in a sentence. The following example demonstrates why she handles them differently.

Example 48

a. Fred$_i$ believes that Jack$_j$ loves himself$_j$.
b. Fred$_i$ believes that Jack$_j$ loves him$_i$.

The clause containing the reflexive pronoun (called R-pronoun by Reinhart) in 48a must also

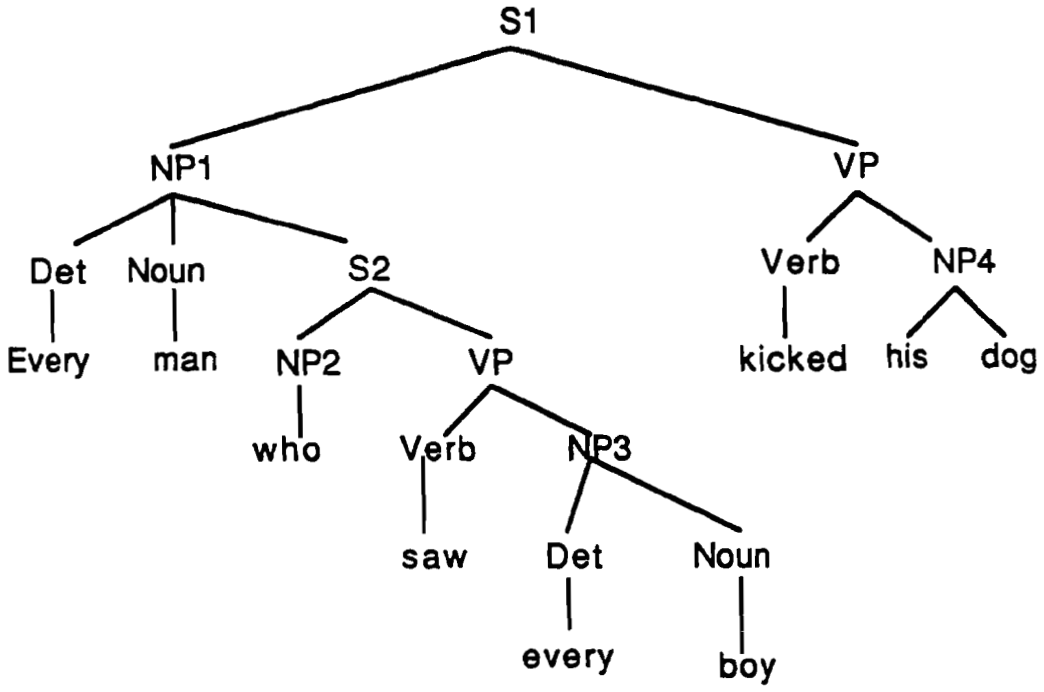

Figure 2.1: Parse tree for *Every man who saw every boy kicked his dog.*

contain the pronoun's antecedent. Non-reflexive pronouns are different than reflexives, as shown in 48b. The antecedent for *him* cannot be *Jack*, hence, *Jack* cannot bind *him*. However, *Fred* can be the antecedent for *him*. Because of examples like 48, Reinhart specifies a rule for determining when a pronoun can be indexed with its antecedent. This rule distinguishes reflexive and non-reflexive pronouns.

> Co-index a pronoun P with a c-commanding NP α (α not immediately dominated by COMP or $\bar{S}$). conditions: (a) If P is an R-pronoun, α must be in its minimal governing category (b) If P is a non-R-pronoun, α must be outside its minimal governing categories. (pp. 158-9)

The minimal governing category of a node α is the S or NP node immediately dominating the node which assigns case (i.e., a V node, or a P node, etc.). If a pronoun cannot be co-indexed with a certain noun phrase, that pronoun cannot be bound by that noun phrase. If a pronoun is coindexed with a noun phrase, and that noun phrase c-commands the pronoun, the pronoun can receive a bound interpretation.

Reinhart's model determines when a pronoun can be coindexed with another noun phrase. Once a pronoun is coindexed with its antecedent, her model also determines whether the pronoun can be bound by its antecedent using only syntactic information. To account for the variety of pronoun behaviors, Reinhart uses the bound-referential dichotomy. Either a pronoun is bound by its antecedent or refers to the same entity as its antecedent. In this way, her approach is similar to many other approaches (e.g., Sag [44], Partee and Bach [38], and Webber [49]). We will discuss this dichotomy further in section 2.4.3.

In the next section, we briefly discuss two approaches that use the semantic representation of a sentence to determine whether a pronoun can have a certain noun phrase as its antecedent.

Semantic approach: Keenan (1974) and Bach and Partee (1980)

Keenan [31] uses the logical structure of a sentence to determine when noun phrases can be antecedents for pronouns. He divides the semantic structure of sentences into functions and arguments. One type of function in English is the verb phrase. An example of an argument is the subject.

Keenan's anaphora rules are based on the function principle. The function principle indicates that an argument cannot depend on its function for its meaning, though the function can depend on its argument. In particular, Keenan uses the function principle to decide when a pronoun cannot have a certain noun phrase as its antecedent. Keenan claims that noun phrases in a function cannot be the antecedent for a pronoun in argument position. Hence, he would predict that the following example is bad given the specified indexing.

Example 49

```
*He_i loves Fred_i.
Argument: He
Function: (loves Fred)
```

Though Keenan's rule is useful, he claims that a complete theory of anaphora requires additional syntactic constraints. However, he believes that syntax would never violate his semantic principle.

Bach and Partee [3] also formulate rules of coreference based on the function principle. They, however, claim that a theory of anaphora can be provided using only the semantic representation of a sentence. In other words, their theory does not use any syntactic information except for that used to provide their semantic representation. However, because the semantic representation of a sentence is provided by the parser, the division between syntax and semantics is fuzzy. In fact, their anaphora rules make similar predictions to Reinhart's rules.

Some of the anaphora rules suggested by Bach and Partee can be used (instead of Reinhart's c-command rule) to determine when a pronoun can be bound by a quantified noun phrase. One principle suggests that if pronoun is an argument in a function, it cannot be bound by a quantifier contained in that function. In other words, a quantifier in a function (or verb phrase) cannot bind a pronoun which is an argument of that function (or the subject). They also consider the complex noun phrase constraint when considering whether a pronoun can be bound by a quantified noun phrase. A pronoun outside of a relative clause cannot be bound by quantified noun phrases contained in that relative clause. Their rules make similar predictions about when a pronoun can be bound by a quantifier as Reinhart's rule. By using Bach and Partee's semantic rules, we would use information available to the parse-tree-to-logical-form mapper. On the other hand, by using Reinhart's rule, we would use information available in the parse tree. In any case, both approaches agree that there are constraints on when a pronoun can be bound by a quantifier and these constraints are determined using only syntactic and sentence-level information.

2.2.4 Summary

We have shown that the behaviors of pronouns range from bound variables on the one hand to constants on the other. These two behaviors do not, however, cover all of the ways that pronouns behave. We have also shown that pronouns refer to syntactic subjects in two different ways. A pronoun can either refer to the subject indirectly (providing a mechanism for deriving the sloppy reading of the elided sentence) or in some other way compatible with the representation of the subject.

Additionally, we have reviewed some linguistic research suggesting that there are syntactic constraints on when a pronoun can be bound by a quantified noun phrase in a sentence. This information will be very useful when we provide a logical form representation for pronouns in the next section. If we are to develop a logical form representation for pronouns consistent with our logical form constraints, we must provide a single representation for a pronoun which can be provided before we know its antecedent and which is consistent with the range of behaviors that pronoun can exhibit. Hence, we must represent a pronoun before deciding whether it should act like a variable, a constant, or something else. Additionally, the logical form for a pronoun must be compatible with all the behaviors the pronoun can adopt, once its antecedent is known. These two requirements are not incompatible, as we will demonstrate in the rest of this chapter. Though we cannot know what a pronoun's antecedent is before representing the pronoun in logical form, we can determine all of the quantified noun phrases in a sentence that can bind the pronoun using only syntactic information. This information will allow us to provide a logical form representation for pronouns, as we will show in the next section.

2.3 Our Representation

2.3.1 Introduction

In this section, we develop a representation for pronouns in logical form. We borrow extensively from past work. Additionally, we make a number of simplifying assumptions about definites and indefinites. Before introducing our representation of pronouns, we summarize those contributions we borrow from past work and our simplifying assumptions.

To derive the logical form for a sentence, we lambda abstract syntactic subjects (following Sag [44], Williams [51], Webber [49], and Partee and Bach [38]). This allows us to distinguish the subject of a sentence from the verb phrase to handle verb phrase ellipsis. The logical roles of noun phrases in a sentence are identified by position in the predicate-argument structure of the sentence's logical form (if this is insufficient, we could use slot-filler notation to indicate logical roles of noun phrases[7]). The logical subject fills the first slot, the logical object the second, and so on.

For purposes of this chapter, we use the following representations for universal noun phrases, indefinite noun phrases, proper nouns, and possessive noun phrases. Following Webber [49], we represent universal noun phrases as universally quantified and restricted variables. Consider example 50.

Example 50

```
Sentence: Every man is happy.
Representation: ∀x: (man x) (happy x)
Meaning: ∀x (if (man x) (happy x))
```

The colon separating the restriction from the universal quantifier in the representation of the sentence is syntactic sugar. The colon in the second line of example 50 expands to give the meaning indicated on the third line. In this chapter, we assume that indefinite noun phrases are represented as existentially quantified and restricted variables. Consider example 51.

Example 51

```
Sentence: A man is happy.
Representation: ∃x: (man x) (happy x)
Meaning: ∃x (and (man x) (happy x))
```

As for universals, the existential's restriction is the proposition following the colon in the representation of the sentence in 51. However, this colon expands differently for the existential, as shown above (in other words, the colon is overloaded). Following Allen [1] and Schubert and Pelletier [46], all quantifiers are placed in the predicate-argument structure for the sentence, allowing us to put off quantifier scoping decisions until later in processing. In this chapter, we do not introduce a general representation for definite noun phrases. We only consider the behaviors of proper nouns and possessive noun phrases. Possessive noun phrases are represented as functions of the possessive nouns (following Webber [49]). Proper nouns are represented as skolem constants (i.e., skolem functions without arguments). Later in this thesis, we will introduce our representation for definites (in Chapter three) and our representation for indefinites (in Chapter four). In Appendix

[7] In fact, in our implementation described in Chapter six, we use slot-filler notation. It is necessary to use slot-filler notation to make certain that the voice of the trigger sentence matches the voice of the elided sentence. However, for the purposes of Chapters two, three, and four, the slot-filler notation is too cumbersome.

B, we provide the syntax and semantics for our logical form.

Our representation of pronouns is similar in spirit to the pronoun representation in Charniak and McDermott [8]. They represent pronouns as unique skolem constants. Later, when the antecedent of a pronoun is known, the pronoun's skolem constant is equated with some value corresponding to its antecedent. By representing a pronoun as a skolem constant before deciding on its antecedent, Charniak and McDermott divide the process of determining the meaning of a sentence into several independent phases. The representation of the basic logical structure of the sentence is provided before pronoun resolution is carried out. This division is consistent with our compactness and modularity constraints. Together, these constraints require that a pronoun be represented before its antecedent is known. However, to obey the formal consistency constraint, a pronoun's initial representation must be compatible with all the ways that pronoun can act. Because a constant is not compatible with a variable, Charniak and McDermott's representation of pronouns violates formal consistency.

To design a logical form for pronouns consistent with our constraints, we must devise a single logical form consistent with all of the possible behaviors of a pronoun in a sentence. If the constraints are met, we can divide the labor of sentence comprehension into several independent modules. We must provide a representation for a pronoun which reflects what we know about the meaning of that pronoun before pragmatic and contextual information is used to determine the antecedent for that pronoun. Thus, we must provide a general representation for a pronoun without utilizing pragmatic information or information provided in other sentences. Because of the range of behaviors pronouns can adopt, we represent them as functions in logical form[8]. A pronoun function is reminiscent of a skolem function. However, a pronoun function is a composite representation. It limits the possible interpretations for the pronoun (we discuss this further in section 2.3.3), though it does not commit us to one particular antecedent. Each pronoun function must have a unique name (supplied by adding a unique number to the pronoun), and its argument list must be specified in a way consistent with our computational constraints. For a pronoun function to be consistent with these constraints, we must specify its argument list using only syntax and sentence-level semantics. Notice that if there is no way to specify the argument list of a pronoun function without using pragmatic information, then the reasons for using logical form are obscured. We must specify the pronoun function's argument list without violating our computational constraints.

We could represent a pronoun as a function of all of the lambda variables and all of the quantified variables in a specific logical form. However, in English, pronouns cannot refer to every noun phrase in a sentence. Consider example 52.

Example 52

*He$_i$ loves every man$_i$.

The antecedent for *he* cannot be *every man*. However, we can determine when certain noun phrases in a sentence can bind a pronoun in the sentence, using only syntactic and sentence-level information. Hence, we can provide a logical form for pronouns which meets our computational constraints. However, we must choose between using Reinhart's rule or Bach and Partee's rule to decide when a pronoun can be bound by a quantified noun phrase in a sentence. Both approaches make similar predictions if we modify Reinhart's c-command rule slightly. Reinhart's rule does not

[8] We provide a single representation for all types of pronouns, though we could represent reflexive, non-reflexive, and possessive pronouns differently. We choose instead to handle differences between these types of pronouns in the algorithm which searches for possible antecedents for a pronoun. This is discussed in Chapter six.

allow a quantified possessive to bind a pronoun, though this is clearly possible, as the following example shows:

Example 53

Every man's$_i$ mother loves him$_i$.

Similarly, a pronoun embedded in a prepositional phrase attached to a noun phrase that c-commands a pronoun can potentially bind that pronoun. For example:

Example 54

A friend of each candidate$_i$ supported him$_i$.

However, as example 55 shows, when a quantified noun phrase is embedded in a relative clause attached to a noun phrase, then that quantifier cannot bind the pronoun even if the containing noun phrase c-commands it.

Example 55

***A man who met each candidate$_i$ supported him$_i$.**

If we allow a pronoun to be bound by a quantified noun phrase that c-commands the pronoun or to be bound by a quantified noun phrase embedded in a noun phrase that c-commands the pronoun but is not affected by the complex noun phrase constraint, then the two approaches give similar results. By using Bach and Partee's semantic rule, we would use information available to the parse-tree-to-logical-form mapper to determine those quantifiers that can bind a pronoun in a sentence. By using our modification of Reinhart's rule, we use information available in the parse tree.

A pronoun function's argument list should also contain the variables of lambda operators corresponding to subjects in order to provide sloppy readings of elided sentences. However, a pronoun function's argument list should only contain the variables of lambda operators that have scope over the position the pronoun function fills in logical form. A lambda operator cannot move. Hence, the position of the lambda operator determines what it can bind. Consider example 56.

Example 56

Fred believes he is happy.

Possible Representation:
Fred$_{22}$, λ(x)(believe x [(he$_1$ x), λ(y)(happy y)])

Because the function for *he* is placed outside of the scope of the lambda operator binding the variable y, that pronoun function cannot be represented as a function of y. The lambda operator for y cannot bind the pronoun because of its position in logical form.

2.3.2 Pronoun Representation

Now that we have provided a way to determine when a pronoun is bound by a quantified noun phrase, we introduce our logical form for pronouns. We represent pronouns as functions in logical form. The argument list of a pronoun function contains those variables associated with noun phrases

that could bind the pronoun. It also contains the variables of lambda operators that have scope over the position the pronoun function fills in logical form (needed to handle potentially sloppy pronouns). Sometimes, a pronoun can have an antecedent which is represented as a quantified variable and is lambda abstracted. Under these circumstances only the lambda variable is included in the argument list since it is the more general argument (the quantified variable is also an argument, given substitution).

Using the bound anaphora rule suggested in the previous section, we propose the following specification for pronoun functions.

The Initial Representation of a Pronoun

> A pronoun is represented as a function in logical form. The name of the function is supplied by appending a unique number to the end of the pronoun string. The pronoun function's argument list consists of all lambda variables (associated with subjects) whose operators have scope over the function in logical form and any variables corresponding to non-subject quantified noun phrases (subjects are already included by virtue of the lambda operator) that c-command the pronoun or corresponding to quantified noun phrases embedded in a noun phrase that c-commands the pronoun but are unaffected by the complex noun phrase constraint.

Because we represent definite noun phrases as functions[9], non-subject definite noun phrases do not affect the representation of pronouns (though a possessive quantified noun phrase which is embedded in a definite noun phrase should affect the representation of pronouns). A pronoun function is an intermediate representation for a pronoun. A pronoun function restricts the possible interpretations (possibly model-theoretic) of a pronoun based on syntax and sentence-level information. However, a pronoun function is not the final meaning for the pronoun. The final meaning cannot be provided until the antecedent of the pronoun is located.

Now that the initial representation of a pronoun (before its antecedent is determined) has been specified, consider a series of examples showing the initial representations of pronouns before their antecedents are known. Consider the representation of the pronoun ***himself*** in example 57.

Example 57

`Fred loves himself.`

Fred_{22}, λ(x)(love x (himself_1 x))

The subject *Fred* is represented as the skolem constant Fred_{22}. The verb phrase is represented as the lambda function, λ(x)(love x (himself_1 x)). Since the sentence in example 57 contains no universal or indefinite noun phrases, the pronoun function representing ***himself*** is simply a function of the lambda variable x. Notice that the name of the pronoun function is the pronoun string concatenated with a unique integer. The only argument in the pronoun function's argument list is the variable x, which is bound by the lambda operator in the representation. However, in the following example, the representation of the pronoun is affected by a universal noun phrase in the sentence.

[9] Actually, possessive noun phrases are represented as functions and proper nouns are represented as skolem constants (though they could be considered as functions without arguments).

Example 58

Fred persuaded every woman that she should go.

```
Fred22, λ(x)(persuade x [∀y: (woman y) y]
                        [(she1 x y), λ(z)(go z)])
```

Notice that the universal quantifier corresponding to *every woman* is placed in the predicate-argument structure of the verb phrase. We do this to avoid choosing a quantifier scope order when a scope ambiguity arises. Because the universal c-commands the pronoun *she*, the universally quantified variable is included in the argument list of the pronoun function, in addition to the lambda variable x. Whenever we create a function whose argument list includes a quantified variable, we require that the quantifier have scope over the function (see Appendix B). Hence, the representation in 58 is equivalent to the representation shown in 59.

Example 59

Fred persuaded every woman that she should go.

```
Fred22, λ(x)(∀y: (woman y)
              (persuade x y [(she1 x y), λ(z)(go z)]))
```

Now consider example 60. In this example, the pronoun is not represented as a function of the universal variable.

Example 60

Fred believes he must speak to every woman.

```
Fred22, λ(x)(believe x  [(he1 x), λ(z)(speak z  [∀y: (woman y) y])])
```

Though the sentence in example 60 contains a universal noun phrase, *he* is represented as a function of the lambda variable x only. The antecedent for *he* cannot be *every woman* because it does not c-command the pronoun. Hence, the pronoun function's argument list does not contain the variable y. Finally, consider an example of a sentence containing only definite noun phrases.

Example 61

Fred showed his mother her picture.

```
Fred22, λ(x)(show x (picture-of (her1 x)) (mother-of (his2 x)))
```

Both of the pronouns in example 61 are represented as functions of the lambda variable x.

Some people may be surprised that we represent reflexive and non-reflexive pronouns in the same way. A uniform representation of pronouns is necessary, even though the type of pronoun limits the set of possible antecedents. For example, a reflexive pronoun's antecedent must be found in the same clause as the pronoun. However, even though the antecedent of a reflexive pronoun must be found in the same clause, its antecedent could be affected by noun phrases outside of the clause. Consider example 62.

Example 62

Every man$_i$ believes that George$_j$ told his$_i$ mother$_k$ about herself$_k$.

The antecedent for the reflexive *herself* must be found within the clause *George told his mother about herself.* However, because the antecedent for the pronoun is subject to outside influences, so must the reflexive. Hence, we provide a uniform representation for pronouns in logical form, regardless of type. We should point out, however, that for correctness, a pronoun resolution module must consider the type of pronoun when determining the set of possible antecedents. In our implementation (described in Chapter six), we use the type of a pronoun to decide which noun phrases are possible antecedents for the pronoun. There are a variety of other factors that affect pronoun resolution (e.g., Hobbs [25,26], Sidner [48], Charniak [4,5], etc.). For a review of the work on anaphora, see Hirst [24]. Though we do not take these factors into account in our representation of a pronoun, these factors are quite important for selecting a proper antecedent for the pronoun.

Once we choose the antecedent for a pronoun, we must indicate the final meaning of the pronoun in logical form. We equate a pronoun function with a value depending on the type of its antecedent. In this way, a pronoun function adopts the behavior of its antecedent. A pronoun is like a chameleon. Depending on the type and location of a pronoun's antecedent, the pronoun function is equated with various values. If a pronoun's antecedent is a universal or indefinite noun phrase in the same sentence, then the pronoun function is set equal to a quantified variable. If a pronoun's antecedent is a noun phrase represented as a function (i.e., a pronoun or definite) in the same sentence, then the pronoun function is set equal to a function. If a pronoun's antecedent is the syntactic subject of a sentence, then the function is equated with either the subject's lambda variable or something depending on the subject's type. If a pronoun is referentially dependent on a noun phrase in a different sentence or on some non-linguistic entity, then the pronoun function is equated with a discourse entity.

When we augment logical form with antecedent information, we must always assert the equality statement in the same lambda environment as the pronoun function to avoid creating logical forms with unbound variables. The initial representation of a pronoun function must be formally consistent with its final meaning. A pronoun function can certainly be equated with a constant, any of its arguments[10], or some pronoun function whose argument list is a subset of the original pronoun function's argument list. Also, a pronoun function can be equated with a possessive function if the possessive function's argument is a discourse entity, a variable contained in the pronoun function's argument list, or pronoun function whose argument list is a subset of the original pronoun function's argument list. If we add pronoun information to the initial logical form in this limited manner, then the logical form will never contain unbound variables after the update. Because we avoid providing logical forms with unbound variables, the initial logical form for a pronoun limits the possible final interpretations for a sentence. We use our initial pronoun representation to determine when a noun phrase is incompatible with the pronoun. However, to provide the final meaning of a sentence, we must also locate the pronoun's antecedent. In this thesis, we assume that a pronoun resolution module exists.

Consider some examples of how logical form is augmented following pronoun resolution. Given that the antecedent for *she* is *every woman* in example 58, the updated logical form is shown in 63.

[10]Or any variables lambda abstracted from the argument list of a pronoun function.

Example 63

Fred$_i$ persuaded every woman$_j$ that she$_j$ should go.

```
Fred22, λ(x)(and (persuade x [∀y: (woman y) y]
                             [(she1 x y), λ(z)(go z)])
                 (= (she1 x y) y))
```

To indicate the fact that the antecedent for *she* is *every man*, the pronoun function (she$_1$ x y) is equated with the universally quantified variable y. Notice that the equality statement is added in the environment of the λ(x) operator. If we added the equality statement outside of this environment, the variable x would have been unbound. Finally, we use equality to substitute y for the function (she$_1$ x y). Hence, the logical form in 63 can be simplified (as shown in 64).

Example 64

Fred$_i$ persuaded every woman$_j$ that she$_j$ should go.

```
Fred22, λ(x)(persuade x [∀y: (woman y) y]
                       [y, λ(z)(go z)])
```

Next, consider how the representation in example 57 would be augmented after pronoun resolution (shown in 65).

Example 65

Fred$_i$ loves himself$_i$.

```
Fred22, λ(x)(and (love x (himself1 x))
                 (or (= (himself1 x) x)
                     (= (himself1 x) Fred22)))
```

Given that the antecedent for *himself* is the subject *Fred*, the pronoun can refer to that pronoun either directly or indirectly. Hence, the pronoun function is equated with either the lambda variable x or Fred$_{22}$. So long as we know what the antecedent of the pronoun is, by allowing a disjunction of equality statements, we can compactly represent the ambiguous ways that pronouns refer to syntactic subjects. In particular, if there are n pronouns whose antecedents are the subject of the sentence, we can specify this ambiguity with O(n) updates (compared with 2^n different representations for the sentence).

Finally, consider how the logical form in example 61 is updated following pronoun resolution (update shown in example 66).

Example 66

Fred$_i$ showed (his$_i$ mother)$_j$ her$_j$ picture.

```
Fred22, λ(x)(and (show x (picture-of (her1 x))
                         (mother-of (his2 x)))
                 (= (her1 x) (mother-of (his2 x)))
                 (or (= (his2 x) x)
                     (= (his2 x) Fred22)))
```

Since the antecedent of *his* is a subject, the pronoun function (his_2 x) is equated with $Fred_{22}$ or x. Since the antecedent of *her* is *his mother*, the pronoun function (her_1 x) is equated with (mother-of (his_2 x)).

We represent pronouns as functions to provide an intermediate level of meaning. For this representation to be useful, it must obey our three computational constraints. Our initial representation of a pronoun as a function is provided using only syntax and local semantics. Hence, our logical form for pronouns satisfies the modularity constraint. Because our initial representation for a pronoun is compatible with all of the possible behaviors of its pronoun, it also satisfies compactness. Finally, because we can update the logical form for a pronoun consistently with its initial meaning, the logical form for a pronoun satisfies the formal consistency constraint. Because a pronoun function can only be updated in certain ways, it limits the possible meanings of its pronoun. This information is useful in a computer model of language comprehension, as we show in Chapter six.

2.3.3 Verb Phrase Ellipsis

In this section, we discuss how our representation of pronouns is used to handle examples of verb phrase ellipsis. Because we assume the meaning of an elided sentence is constrained by the meaning of its trigger sentence, the meaning of a trigger sentence must be specified before the meaning of the elided sentence can be provided. We illustrate our approach to verb phrase ellipsis with example 6.

Example 6

```
Trigger Sentence: Fredi loves hisi wife.
Elided Sentence:  Georgej does too.
Meanings:
1. George loves Fred's wife.
2. George loves George's wife.
```

To determine the meaning of the elided sentence in this example, we must decide on a meaning for the trigger sentence. In particular, we must determine the antecedent for the pronoun *his*.

The trigger sentence in example 6 is initially represented as follows:

Example 67

```
Fred loves his wife.

Fred22, λ(x)(love x (wife-of (his1 x)))
```

Before this representation of the trigger sentence is used to provide the meaning of the elided verb phrase in example 6, we must locate the antecedent for *his*. Otherwise, the final meaning of

the pronoun function in the trigger cannot limit the meaning of the pronoun in the elided sentence. Assume that the antecedent for *his* is *Fred*. Because *Fred* is the subject, the pronoun *his* can refer to it either directly (by equating the pronoun function with the subject's discourse entity) or indirectly (by equating the pronoun function with the subject's lambda variable). Since we have no way to choose between these two possible meanings of the pronoun, we augment the logical form as shown in 68.

Example 68

Fred$_i$ loves his$_i$ wife.

```
Fred22, λ(x)(and (love x (wife-of (his1 x)))
                 (or (= (his1 x) x)
                     (= (his1 x) Fred22)))
```

Since *his* refers to the subject, there are two ways to update the logical form. This ambiguity is indicated by the disjunction in logical form. This logical form compactly stores two possible representations of the trigger sentence.

Next, consider the initial representation of the elided sentence in 6. To represent the elided sentence, we introduce a dummy predicate to indicate the fact that there is a missing verb phrase.

Example 69

George does too.

```
George35, λ(y)(Dummy2 y)
```

Since the elided verb phrase must be represented in some manner, we represent it like any other verb phrase (except for the predicate). However, this representation is simply a place holder which must be replaced by the trigger verb phrase once it is located. In other words, the dummy function in 69 must be replaced with the representation of the trigger verb phrase once a single meaning for the trigger sentence is picked.

To determine the meaning of the elided verb phrase in 69, we must pick one meaning for the pronoun function in 68[11]. We do not consider how the choice between the two meanings is decided, we simply demonstrate that for each disjunct, we can derive a different meaning for the elided sentence. If we assume that the pronoun *his* refers indirectly to the subject, then the pronoun function (his$_1$ x) is equated with (replaced by) the lambda variable x. This choice allows us to provide the sloppy reading of the elided sentence, as shown in 70.

[11]Making this choice requires contextual and pragmatic information.

Example 70

```
Trigger Sentence Representation:
Fred22, λ(x)(and (love x (wife-of (his1 x)))
                 (= (his1 x) x))
Elided Sentence Representation:
George35, λ(y)(Dummy2 y)
Replace λ(y)(Dummy2 y) with
 λ(x)(and (love x (wife-of (his1 x)))
          (= (his1 x) x))
to get:
George35, λ(x)(and (love x (wife-of (his1 x)))
                   (= (his1 x) x))
```

Given that the pronoun *his* refers directly to the subject *Fred*, the other update is chosen. This allows us to derive the strict reading of the elided sentence, as shown in 71.

Example 71

```
Trigger Sentence Representation:
Fred22, λ(x)(and (love x (wife-of (his1 x)))
                 (= (his1 x) Fred22))
Elided Sentence Representation:
George35, λ(y)(Dummy2 y)
Replace λ(y)(Dummy2 y) with
 λ(x)(and (love x (wife-of (his1 x)))
          (= (his1 x) Fred22))
to get:
George35, λ(x)(and (love x (wife-of (his1 x)))
                   (= (his1 x) Fred22))
```

Once it is possible to decide on a single meaning for the trigger sentence, we can derive the appropriate reading of the elided sentence. We have demonstrated how we use a dummy function to initially represent the missing verb phrase in the elided sentence, and how this dummy function is replaced with the representation of the trigger verb phrase to provide the meaning of the elided sentence. We omit this step in most of the examples of this thesis. We include this step in our discussion to emphasize the fact that the null verb phrase in an elided sentence is an anaphor whose meaning must be determined.

The reader might suggest that there is no need to pick the appropriate representation of the trigger sentence before deriving the meaning of the elided sentence, since the point of ambiguity resides with the the elided sentence not with the trigger sentence. This, however, is not necessarily true. There are sentences outside of verb phrase ellipsis in which pronouns receive two different meanings. Consider example 72.

Example 72

```
Only George voted for himself.
Meanings:
1. George was the only person to vote for George.
   ∀x: (≠ x George) (and George, λ(y)(vote y George)
                          ¬ [x, λ(y)(vote y George)])
2. George was the only person to vote for himself.
   ∀x: (≠ x George) (and George, λ(y)(vote y y)
                          ¬ [x, λ(y)(vote y y)])
```

In the first meaning of the sentence in 72, ***himself*** is replaced by *George*. However, in the second meaning, the pronoun is replaced by the lambda variable *y*. Hence, selecting the meaning of the trigger sentence before deriving the meaning of the elided sentence seems reasonable.

The method we use to provide interpretations for elided sentences consists of many phases. First, we provide the initial representation for a sentence. Any sentence we process could be a trigger sentence. There is no way to know until we come across an elided sentence. When an elided sentence is encountered, its trigger sentence must be located. When a trigger sentence is located, the meaning of the sentence must be specified. We must determine the antecedents for any pronouns in the sentence and update the initial logical form for the sentence. We must also pick one reading in case a pronoun refers to its antecedent in two different ways. Once the appropriate meaning of the trigger sentence is selected, we use the trigger verb phrase to provide the meaning of the elided verb phrase. As the reader may notice, there are a large number of steps in this model requiring pragmatic information. We can provide the initial representation of the trigger or elided sentences without using pragmatics. However, pragmatic and contextual information is necessary to pick the intended meaning of the trigger sentence. This information is used to update logical form and pick the appropriate meaning in the face of ambiguity.

In this chapter, we discussed only very simple examples of the sloppy identity ambiguity (i.e., examples where the antecedent for a single pronoun in a trigger sentence is the subject of that trigger sentence). Example 6 is such an example:

Example 6

```
Trigger Sentence: Fred_i loves his_i wife.
Elided Sentence:  George_j does too.
Meanings:
1. George loves Fred's wife.
2. George loves George's wife.
```

When the subject of the trigger sentence is the antecedent for a single pronoun in the same sentence, two meanings of the elided sentence result. In 6, two readings result when the antecedent for ***his*** is *Fred*, because the pronoun can refer to that noun phrase either directly or indirectly. Does this mean that when the antecedent for two pronouns is the subject of a trigger sentence, then four meanings of an elided sentence result? Certainly, in our approach, we would generate four possible meanings. However, at least in some situations, not all of the four meanings for the elided sentence are reasonable. Sag [44] discusses an example of verb phrase ellipsis (i.e., example 73) in which the trigger sentence contains two pronouns whose antecedent is the subject of the sentence (the example was first noticed by Dahl [15]).

Example 73

Bill$_i$ believed he$_i$ loved his$_i$ wife.
Harry$_j$ did too.
Meanings:
1. Harry believed that Harry loved Harry's wife.
2. Harry believed that Bill loved Bill's wife.
3. Harry believed that Harry loved Bill's wife. (marginal)
4. *Harry believed that Bill loved Harry's wife.

Despite the fact that Sag's model predicts the elided sentence should have four possible meanings, it has only three reasonable meanings. Our approach suffers from the same problem. In Chapter five, we will discuss a constraint proposed by Sag to handle example 73. We will also discuss a constraint we propose to handle additional examples of multiple pronoun verb phrase ellipsis.

2.4 Previous Pronoun Representations

In this section we review previous representations of pronouns. Though each approach discussed here uses the bound-referential dichotomy to model pronoun behavior, each approach is different. Hence, some examples present problems for one approach but not for the others.

2.4.1 Sag (1976)

In his approach to verb phrase ellipsis, Sag [44] represents a pronoun as a string with an index indicating its antecedent. He defines a rule to generate all possible representations of a trigger sentence when a pronoun is co-indexed with a syntactic subject. This rule specifies that a pronoun co-indexed with a subject can *optionally* be replaced by the lambda variable associated with the subject. This rule is shown in 74.

Example 74

Pro -> BV Rule

NP_i, $\lambda(x)(\ldots Pro_i \ldots)$
$\Downarrow$
x

This rule takes a logical form in which all pronouns are indexed with their antecedents. It generates additional representations for a sentence whenever a pronoun in the logical form is coindexed with the subject. For example, given the logical form $Fred_i$, $\lambda(x)$(love x himself$_i$), the rule generates an additional logical form for the sentence (i.e., $Fred_i$, $\lambda(x)$(love x x)). To use this rule, Sag assumes that indices (indicating coreference between noun phrases) are assigned to all noun phrases in the trigger sentence, including non-referential noun phrases like *everyone*. Sag claims that indexed pronouns are referential, unless they are co-indexed with quantified noun phrases. He defines a rule to *obligatorily* replace a pronoun indexed with a quantified noun phrase with the quantified variable. Once all of the pronouns coindexed with quantified noun phrases are replaced by variables, the remaining pronouns with indices are referential.

Sag represents the trigger sentence in Example 6 initially by co-indexing the pronoun with its antecedent (see example 75a).

Example 6

Trigger Sentence: Fred$_i$ loves his$_i$ wife.
Elided Sentence: George$_j$ does too.
Meanings:
1. George loves Fred's wife.
2. George loves George's wife.

With his optional rule, he derives the second representation by replacing the pronoun string with the subject's lambda variable (see 75b).

Example 75

Fred$_i$ loves his$_i$ wife.

a. Fred$_i$, λ(x)(loves x his$_i$ wife)
; Fred loves Fred's wife
b. Fred$_i$, λ(x)(loves x x's wife)
; Fred loves his own wife

Each representation in 75 sanctions a different meaning for the elided sentence *George does too*, as we have already shown in Chapter one.

Sag's representation of pronouns does not satisfy our logical form constraints. Because he provides a representation for a pronoun only after its antecedent is known, his pronoun representation violates the compactness and modularity constraints. Sag's optional pronoun rule handles the ambiguity found in example 6. However, replacing a co-indexed pronoun string with a variable violates formal consistency. A pronoun string is not compatible with a variable. Additionally, Sag's optional rule generates 2^n distinct representations for a trigger sentence containing n pronouns that refer to its syntactic subject. Generating this many representations of a trigger sentence violates compactness. A more compact way to represent this ambiguity is needed. Sag's model of verb phrase ellipsis handles a variety of examples though he was not concerned with computational issues. However, because we want to provide a model of language comprehension consistent with our logical form constraints, we cannot use Sag's approach.

Sag's representation of pronouns has additional problems. In particular, he assumes that pronouns co-indexed with definite noun phrases are referential. Example 76 shows the error of this assumption.

Example 76

Every boy$_i$ showed (his$_i$ mother)$_j$ her$_j$ clock.

Sag could represent this sentence, as shown in 77.

Example 77

Every boy$_i$ showed (his$_i$ mother)$_j$ her$_j$ clock.

$\forall$x: (boy x) x, λ(y)(show y (x's mother)$_j$ her$_j$ clock)

Notice that *her* is coindexed with (x's mother). Sag assumes that indices are assigned to all pronouns and their antecedents. Whenever a pronoun is coindexed with a quantified noun phrase, it is replaced by that variable. After this replacement, all coindexed pronouns are assumed to be referential. However, since (x's mother) is not quantified, *her* cannot be replaced by a variable. Also, since (x's mother) is not referential, *her* is not referential. Hence, the index on *her* and (x's mother) is not defined in Sag's model.

Because of the way we represent pronouns and possessives, our model captures the correct meaning of the sentence in example 76. Before pronoun resolution, the sentence in example 76 is represented as shown in 78.

Example 78

```
Every boy showed his mother her clock.

∀x: (boy x) x, λ(y)(show y (clock-of (her1 y)) (mother-of (his2 y)))
```

Given that the antecedent for *his* is *every boy* and the antecedent for *her* is *his mother*, our initial representation is augmented, as shown in 79.

Example 79

```
Every boy_i showed (his_i mother)_j her_j clock.

∀x: (boy x) x, λ(y)(and (show y (clock-of(her1 y)) (mother-of (his2 y)))
                        (or (= (his2 y) y)
                            (= (his2 y) x))
                        (= (her1 y) (mother-of (his2 y))))
```

Given that we pick one of the meanings for (his_2 y) in 79, we are able provide a reasonable meaning for the sentence in example 76.

Now consider how Sag's model handles a similar example in verb phrase ellipsis. Consider example 80.

Example 80

```
Trigger Sentence: Fred_i showed (his_i mother)_j her_j dog.
Elided Sentence:  George_l did too.
Meanings:
1. George showed Fred's mother Fred's mother's dog.
2. George showed George's mother George's mother's dog.
3. *George showed George's mother Fred's mother's dog.
4. *George showed Fred's mother George's mother's dog.
```

Given the indices on the noun phrases, the elided sentence has two meanings, that is the first and second meaning shown in example 80. However, Sag's approach sanctions the first and third meaning indicated in the example.

Sag's initial representation of the trigger sentence is shown in 81a.

Example 81

Fred$_i$ showed (his$_i$ mother)$_j$ her$_j$ dog.

a. Fred$_i$, λ(x)(showed x (her$_j$ dog) (his$_i$ mother)$_j$)
b. Fred$_i$, λ(x)(showed x (her$_j$ dog) (x's mother))

The optional rule to convert a pronoun to a bound variable is applied to his$_i$, giving the other possible representation of the trigger sentence (i.e., 81b)[12]. The candidate sentence for verb phrase ellipsis is *George showed his mother her dog*, which can be represented in a variety of ways, as shown in 83.

Example 83

George showed his mother her dog.

a. George$_l$, λ(y)(showed y (her$_j$ dog) (his$_i$ mother)$_j$)
   ; George showed Fred's mother Fred's mother's dog
b. George$_l$, λ(y)(showed y (her$_m$ dog) (his$_l$ mother)$_m$)
   ; George showed George's mother George's mother's dog
c. George$_l$, λ(y)(showed y (her$_m$ dog) (y's mother))
   ; George showed George's mother George's mother's dog
d. George$_l$, λ(y)(showed y (her$_j$ dog) (y's mother))
   ; George showed George's mother Fred's mother's dog
e. George$_l$, λ(y)(showed y (her$_n$ dog) (his$_i$ mother)$_j$)
   ; George showed Fred's mother some other woman's dog
f. George$_l$, λ(y)(showed y (her$_p$ dog) (y's mother))
   ; George showed George's mother some other woman's dog
g. George$_l$, λ(y)(showed y (her$_s$ dog) (his$_r$ mother)$_s$)
   ; George showed another man's mother that mother's dog
h. George$_l$, λ(y)(showed y (her$_w$ dog) (his$_u$ mother)$_v$)
   ; George showed another man's mother some other woman's dog

The only sanctioned meanings for the elided verb phrase are alphabetic variants of the trigger representations in 81. Since the verb phrases in 83a and 81a are alphabetic variants, the elided sentence can receive the meaning *George showed Fred's mother Fred's mother's dog.* However, because the verb phrase in 83d is an alphabetic variant of the verb phrase in 81b, the elided sentence can also mean *George showed George's mother Fred's mother's dog.* This reading is not reasonable. Additionally, a very good reading, shown in 83c, is not sanctioned[13].

On the other hand, our model provides reasonable meanings for example 80. The trigger sentence

[12] We could represent 81b as follows:

Example 82
Fred$_i$, λ(x)(showed x (her$_j$ dog) (x's mother)$_j$)

However, what does her$_j$ mean since (x's mother) is not quantified or referential?

[13] We might be able to provide the sloppy reading for the elided sentence by using the representation in 82 to represent the trigger sentence. Then we might represent the elided sentence as follows:

Example 84
George$_l$, λ(y)(showed y (her$_j$ dog) (y's mother)$_j$)

Though the verb phrase in 84 is an alphabetic variant of the verb phrase in 82, the representation is very odd. The index j indicates that George's mother is coreferential with Fred's mother.

is initially represented as shown in 85.

Example 85

```
Fred showed his mother her dog.

Fred22, λ(x)(show x (dog-of (her1 x)) (mother-of (his2 x)))
```

Given that the antecedent for *his* is *Fred*, the pronoun function can be equated with $Fred_{22}$ or the lambda variable x. Likewise, because the antecedent for *her* is *his mother* (which is not a subject), the pronoun function for *her* is equated with the function representing that noun phrase. The logical form of the trigger sentence after pronoun resolution is shown in 86.

Example 86

```
Fredi showed (hisi mother)j herj dog.

Fred22, λ(x)(and (show x (dog-of (her1 x)) (mother-of (his2 x)))
                 (or (= (his2 x) x)
                     (= (his2 x) Fred22))
                 (= (her1 x) (mother-of (his2 x))))
```

The representation of the trigger sentence in 86 contains two different representations of the trigger sentence. Each representation allows us to derive a different meaning of the elided sentence.

Suppose that we decide that *his* refers directly to the subject *Fred*, then we would pick the second disjunct in 86 as the intended meaning of the pronoun. Hence, we are able to provide the strict reading of the elided sentence, as shown in 87.

Example 87

```
Trigger Sentence Representation:
Fred22, λ(x)(and (show x (dog-of (her1 x)) (mother-of (his2 x)))
                 (= (his2 x) Fred22)
                 (= (her1 x) (mother-of (his2 x))))
Elided Sentence Representation:
George2, λ(x)(and (show x (dog-of (her1 x)) (mother-of (his2 x)))
                  (= (his2 x) Fred22)
                  (= (her1 x) (mother-of (his2 x))))
; George showed Fred's mother Fred's mother's dog.
```

Suppose that we decide that *his* refers indirectly to the subject, then we would pick the first disjunct in 86 as the intended meaning of the pronoun. Hence, we are also able to provide the sloppy reading of the elided sentence.

Example 88

```
Trigger Sentence Representation:
Fred22, λ(x)(and (show x (dog-of (her1 x)) (mother-of (his2 x)))
                 (= (his2 x) x)
                 (= (her1 x) (mother-of (his2 x))))
Elided Sentence Representation:
George2, λ(x)(and (show x (dog-of (her1 x)) (mother-of (his2 x)))
                  (= (his2 x) x)
                  (= (her1 x) (mother-of (his2 x))))
; George showed George's mother George's mother's dog.
```

Because we equate the pronoun function for *her* with the possessive function for *his mother*, we can derive the expected readings of the elided sentence of example 80.

As demonstrated by examples 76 and 80, Sag's approach does not model pronouns whose antecedents are non-referential definites. One way to fix this problem would be to make definite noun phrases quantificational. However, if we represent definites as quantified terms, other problems arise. By representing a definite noun phrase as a function, and by allowing a pronoun to adopt the function of its antecedent as its meaning, we provide a solution that bridges the gap between the representation of definites as quantified terms and the representation of definites as constants.

2.4.2 Webber (1978)

Webber [49] initially represents a pronoun as a string in logical form. The pronoun string is replaced by a pronoun trace equated with something depending on the type of the pronoun's antecedent[14]. An example of a pronoun trace is shown in example 89b (see example below). What a pronoun trace is equated with depends on the type of its antecedent. If the pronoun's antecedent is a noun phrase represented as a quantified variable, the pronoun trace is equated with that variable. If the pronoun's antecedent is a definite noun phrase, the pronoun trace is equated with a discourse entity. A discourse entity uniquely identifies an individual denoted by a certain noun phrase in the discourse model of the speaker/hearer. Like Sag, Webber defines an optional rule to derive an additional representation for a trigger sentence containing a pronoun that refers to its subject. This rule replaces the pronoun trace with the lambda variable associated with the subject.

Consider how Webber's approach handles example 6.

Example 6

```
Trigger Sentence: Fred_i loves his_i wife.
Elided Sentence:  George_j does too.
Meanings:
1. George loves Fred's wife.
2. George loves George's wife.
```

Example 89 shows Webber's representations for the trigger sentence in example 6.

[14] Webber uses a pronoun trace to identify pronouns in the logical form representation of a sentence since the sloppy identity ambiguity is possible only for pronouns.

Example 89

Fred$_i$ loves his$_i$ wife.

a. Fred, λ(r)(love r (wife-of (his)))
b. Fred, λ(r)(love r (wife-of (Pro = Fred_{22})))
c. Fred, λ(r)(love r (wife-of (r)))

The initial representation of *his* is shown in 89a. Notice that the antecedent for *his* has not yet been determined. Once pronoun resolution occurs, the pronoun string is replaced by the pronominal trace representation (shown in 89b). Finally, the bound variable interpretation (shown in 89c) is derived from the pronominal trace representation. Webber uses the representations in 89b and 89c to derive the two possible meanings of the elided sentence (obtained by applying the elided sentence's subject to the two lambda functions).

Example 90

George does too.

a. George, λ(r)(love r (wife-of (Pro = Fred_{22})))
   ; George loves Fred's wife
b. George, λ(r)(love r (wife-of (r)))
   ; George loves George's wife

Webber's use of logical form suffers from many of the same problems that Sag's approach does. Replacing a pronoun string by a trace equated with a variable violates formal consistency. Replacing a pronoun trace equated with a discourse entity by a lambda variable also violates the formal consistency constraint. Each augmentation is incompatible with the previous representation of a pronoun. Webber's model also violates compactness. She provides two different representations of a trigger sentence to handle the sloppy identity ambiguity.

Webber [49] assumes that when a pronoun's antecedent is a non-subject definite noun phrase, the pronoun should be replaced by a pronoun trace equated with the antecedent's discourse entity (or referent). Because of this, her approach suffers from the same difficulty with example 76 as Sag's approach does.

Example 76

Every boy$_i$ showed (his$_i$ mother)$_j$ her$_j$ clock.

Following pronoun resolution, the pronoun *his* can be represented as either a lambda variable or a pronoun trace equated with a universally quantified variable. However, the representation of the pronoun *her* presents a problem. The antecedent for *her* is *his mother*. However, since *his mother* is not quantified or referential (i.e., it does not denote a specific mother (or set of mothers) within the sentence), Webber is unable to represent the meaning of *her*.

A related problem arises in example 80.

Example 80

```
Trigger Sentence: Fred_i showed (his_i mother)_j her_j dog.
Elided Sentence:  George_i did too.
Meanings:
1. George showed Fred's mother Fred's mother's dog.
2. George showed George's mother George's mother's dog.
3. *George showed George's mother Fred's mother's dog.
4. *George showed Fred's mother George's mother's dog.
```

Since the antecedent of *his* is *Fred*, the pronoun is represented in two ways. Because the antecedent of *her* is *his mother*, which is not a syntactic subject, the pronoun trace must be equated with the discourse entity for its antecedent (say mother_{22}). Thus, Webber's model provides two representations of the trigger sentence, shown in 91.

Example 91

Fred$_i$ showed (his$_i$ mother)$_j$ her$_j$ dog.

a. Fred, λ(x)(show x (dog-of (Pro = mother_{22}))
(mother-of (x)))
b. Fred, λ(x)(show x (dog-of (Pro = mother_{22}))
(mother-of (Pro = Fred_{22})))

Each representation of the trigger sentence indicates the meaning of the trigger sentence. However, one of the two derived meanings of the elided sentence (indicated below) is not an expected reading.

Example 92

George did too.

a. George, λ(x)(show x (dog-of (Pro = mother_{22}))
(mother-of (x)))
; George showed George's mother Fred's mother's dog.
b. George, λ(x)(show x (dog-of (Pro = mother_{22}))
(mother-of (Pro = Fred_{22})))
; George showed Fred's mother Fred's mother's dog.

The meaning for the elided sentence in 92b is reasonable, but the one in 92a is not. Moreover, one of the expected meanings (i.e., the second meaning in example 80) cannot be derived.

In summary, Webber's model of verb phrase ellipsis does not obey our computational constraints. Additionally, her approach does not model the behavior of pronouns whose antecedents are non-referential definite noun phrases.

2.4.3 Reinhart (1983)

Reinhart [40] claims that pronouns behave in two different ways, either as bound anaphors or as coreferential pronouns. Bound anaphora corresponds with the bound variable representation of a pronoun suggested by Sag [44] and Webber [49]. Coreference, on the other hand, corresponds with Webber's discourse entity.

As we already discussed in section 2.3.2, Reinhart provides a syntactic rule for determining when a

pronoun can be bound by its antecedent. A pronoun can be bound by its antecedent if and only if that pronoun is c-commanded by its antecedent in the parse tree. Reinhart claims that pronouns can be bound variables regardless of whether their antecedents are quantified or not. For example, Reinhart does not represent pronouns or definites as quantified terms. Yet, she claims that when a pronoun's antecedent is a definite noun phrase or a pronoun, and the antecedent c-commands the pronoun, then the pronoun receives a bound interpretation. Reinhart use lambda abstraction to provide the necessary bound interpretation. There are, however, a number of difficulties with Reinhart's solution.

At first glance, the idea of binding a pronoun with the lambda operator of its antecedent (given that the noun phrase c-commands the pronoun) seems promising. Consider example 76.

Example 76

Every boy$_i$ showed (his$_i$ mother)$_j$ her$_j$ clock.

By lambda abstracting the noun phrase *his mother*, it is possible to provide a reasonable representation for the meaning of the sentence (shown in 93).

Example 93

Every boy$_i$ showed (his$_i$ mother)$_j$ her$_j$ clock.

$\forall$x: (boy x) x, λ(y)(y's mother, λ(z)(show y z's clock z)

However, in English, a non-referential definite can be a pronoun's antecedent even if it does not c-command the pronoun. Consider example 94.

Example 94

Every man$_i$ discussed the behavior of (his$_i$ mother)$_j$ with her$_j$ psychiatrist.

In this example, the antecedent of *her* is *his mother*. However, *her* cannot denote a specific individual (because *his* refers to *every man*, and acts like a bound variable). Additionally, the noun phrase *his mother* cannot bind the pronoun *her* (in Reinhart's approach) since it does not c-command the pronoun. Though Reinhart can handle example 76, she cannot handle any example where a pronoun has an antecedent which neither refers to a specific individual within the sentence[15] nor c-commands the pronoun. In Chapter three, we will introduce a general representation for definite noun phrases. Once we do, we will demonstrate how our approach can handle this example.

Reinhart's approach also fails to handle examples of verb phrase ellipsis in which a non-referential definite is the antecedent for a pronoun it does not c-command. Consider example 95.

[15]This happens most notably when a pronoun in the definite noun phrase refers directly or indirectly to a universal noun phrase.

Example 95

Fred$_i$ discussed the behavior of (his$_i$ mother)$_j$ with her$_j$ psychiatrist.
Bill$_k$ did too.
Meanings:
1. Bill discussed the behavior of Bill's mother with Bill's mother's psychiatrist.
2. Bill discussed the behavior of Fred's mother with Fred's mother's psychiatrist.
3. *Bill discussed the behavior of Bill's mother with Fred's mother's psychiatrist.
4. *Bill discussed the behavior of Fred's mother with Bill's mother's psychiatrist

Because *his mother* does not c-command *her*, Reinhart's model can provide the two representations of the trigger sentence, shown in 96.

Example 96

Fred$_i$ discussed the behavior of (his$_i$ mother)$_j$ with her$_j$ psychiatrist.

a. Fred, λ(x)(discuss-with x [the behavior of x's mother]
[mother22's psychiatrist])
b. Fred, λ(x)(discuss-with x [the behavior of Fred's mother]
[mother22's psychiatrist])

Using these two trigger representations, two representations for the meaning of the elided sentence, shown in 97, are derived.

Example 97

Bill did too.

a. Bill, λ(x)(discuss-with x [the behavior of x's mother]
[mother22's psychiatrist])
; Bill discussed the behavior of Bill's mother with Fred's mother's
; psychiatrist
b. Bill, λ(x)(discuss-with x [the behavior of Fred's mother]
[mother22's psychiatrist])
; Bill discussed the behavior of Fred's mother with Fred's mother's
; psychiatrist

Reinhart is unable to derive the best reading of the elided sentence, namely *Bill discussed the behavior of Bill's mother with Bill's mother's psychiatrist.* Hence, Reinhart's approach does not properly describe the way pronouns act in verb phrase ellipsis. In Chapter three, we will demonstrate how our approach handles this example.

Consider Reinhart's use of the lambda operator to bind pronouns that refer intrasententially to definite noun phrases. Temporarily ignore the c-command issue. If a definite antecedent always binds a pronoun with a lambda operator, then there should only be one meaning for a pronoun whose antecedent is a non-referential definite subject. However, consider example 98[16].

[16]The inspiration for this example was a similar example discussed in Sells, Zaenen, and Zec [47].

Example 98

Every man$_i$ believes that (his$_i$ wife)$_j$ can defend herself$_j$ better than he$_i$ can.
Meanings:
1. Every man believes that his wife can defend herself better than he can defend himself.
2. Every man believes that his wife can defend herself better than he can defend her.

Reinhart's approach provides only one way for a definite noun phrase to bind a pronoun. She can provide the first meaning of the sentence in 98 by replacing the pronoun *herself* with the lambda variable corresponding to ***his wife***. However, she cannot provide the second reading[17] since *herself* must be either lambda bound or referential in her model. To provide the second reading, the pronoun *herself* could be bound by some quantifier used to represent ***his wife***. However, there are difficulties associated with representing definites as quantified terms. The noun phrase ***his wife*** could be represented as a function. In fact, the second reading of the sentence in 98 can be provided if we replace ***herself*** with the function representing ***his wife***.

By representing definites as functions, our approach provides the two meanings of the elided sentence in example 98. The initial representation for the sentence in this example is shown in 99.

Example 99

Every man believes that his wife can defend herself better than he can.

$\forall$x: (man x)
 x, λ(y)(believe y [(better-than
 (wife-of (his$_1$ y)), λ(z)(defend z (herself$_2$ y z))
 (he$_3$ y), λ(w)(Dummy$_4$ w))])

Notice that we do not provide a representation for ***better than***, since this would be beyond the scope of this work. However, we assume that it relates the two sentences in example 98 in the way indicated in the representation of the sentence. We must determine the antecedents for the pronouns in this example, before settling on the meaning for the elided verb phrase. Given that *his* and *he* refer to *every man* and *herself* refers to *his wife*, the logical form would be updated, as shown in 100.

[17] This example would also be a problem for Sag [44] and Webber [49].

Example 100

Every man$_i$ believes that (his$_i$ wife)$_j$ can defend herself$_j$ better than he$_i$ can.

```
∀x: (man x)
 x, λ(y)(and (believe y [(better-than
                          (wife-of (his1 y)),
                            λ(z)(and (defend z (herself2 y z))
                                     (or (= (herself2 y z)
                                            (wife-of (his1 y)))
                                         (= (herself2 y z) z)))
                          (and (he3 y), λ(w)(Dummy4 w)
                               (or (= (he3 y) x)
                                   (= (he3 y) y)))))])

            (or (= (his1 y) y)
                (= (his1 y) x)))
```

Now given that the trigger verb phrase is the lambda function headed by $\lambda(z)$, we can derive the two readings of the elided sentence. Suppose we decide that *herself* refers directly to *his wife*, then we can derive the strict reading of the elided sentence, as shown in 101.

Example 101

Every man$_i$ believes that (his$_i$ wife)$_j$ can defend herself$_j$ better than he$_i$ can.

```
∀x: (man x)
 x, λ(y)(and (believe y [(better-than
                          (wife-of (his1 y)),
                            λ(z)(and (defend z (herself2 y z))
                                     (= (herself2 y z)
                                        (wife-of (his1 y))))
                          (and (he3 y),
                                 λ(z)(and (defend z (herself2 y z))
                                          (= (herself2 y z)
                                             (wife-of (his1 y))))
                               (or (= (he3 y) x)
                                   (= (he3 y) y)))))])
             (or (= (his1 y) y)
                 (= (his1 y) x)))
 ; Every man believes his wife can defend herself better than he can
 ; defend her.
```

On the other hand, if we assume that *herself* refers indirectly to the subject, then we can derive the sloppy reading, as shown in 102.

Example 102

```
Every man_i believes that (his_i wife)_j can defend herself_j better than
he_i can.

∀x: (man x)
 x, λ(y)(and (believe y [(better-than
                          (wife-of (his_1 y)),
                            λ(z)(and (defend z (herself_2 y z))
                                     (= (herself_2 y z) z))
                          (and (he_3 y),
                                 λ(z)(and (defend z (herself_2 y z))
                                          (= (herself_2 y z) z))
                               (or (= (he_3 y) x)
                                   (= (he_3 y) y))));])
             (or (= (his_1 y) y)
                 (= (his_1 y) x)))
; Every man believes his wife can defend herself better than he can
; defend himself.
```

Hence, we are able to derive the two meanings[18] for the elided sentence in example 98. And, we do so without representing definites as quantified terms.

In summary, Reinhart's approach is not compatible with our computational constraints on logical form. Her representation of a pronoun depends on what the antecedent of the pronoun is, violating compactness and modularity. Additionally, she cannot locally store the ambiguous way that pronouns refer to subjects to handle the sloppy identity ambiguity (violating compactness). To be fair, Reinhart was only interested in characterizing the ways that pronouns act by using syntactic information. She claims that pronouns act as bound variables or constants. In particular, when a pronoun refers to a definite noun phrase that c-commands the pronoun, the pronoun can be bound by the definite (by using lambda abstraction) or can be coreferential with the definite. However, example 98 suggests that pronouns can refer to definite noun phrases in two non-static ways. Lambda abstraction can account for only one of these ways. Additionally, a non-referential definite can be a pronoun's antecedent even though it does not c-command the pronoun.

2.4.4 Partee and Bach (1981)

As discussed in Chapter one, Partee and Bach [38] represent pronouns as variables that are either bound by an operator in the representation of the sentence or are assigned a referent by the context assignment function. They attempt to directly provide the meaning of an elided sentence. However, because they insist on a compositional semantics, they cannot ensure that the meaning of a pronoun in the trigger sentence will affect the meaning of the pronoun in the elided sentence. In particular, a pronoun variable, bound in the trigger sentence, can become unbound in the elided sentence. Because of this, it is possible to obtain very odd readings for the elided sentence, as was already shown in Chapter one. They could require that pronouns that are free in the trigger sentence must remain free in the elided sentence, or pronouns bound by an operator in the trigger sentence must be bound by that operator in the elided sentence. However, this assumption is not compatible with the spirit of their research.

[18] Notice that we have to decide on the meaning of (his_1 y) and (he_3 y) before the final meaning of the sentence is available.

Partee and Bach represent definites as quantified terms. Because of their choice, certain difficulties arise. Consider example 103.

Example 103

```
The boy_i loves his_i dog.
The man_j does too.
```

The trigger sentence can be represented using a quantifier, such as ∃!. There are three ways to represent the pronoun, as shown in 104.

Example 104

```
The boy loves his dog.

a. ∃!x (and (boy x)
            x, lambda(y)[(∃! z) (and (= (dog-of x) z) (love y z))]
   ; his is represented as the bound variable x
b. ∃!x (and (boy x)
            x, lambda(y)[(∃! z) (and (= (dog-of y) z) (love y z))]
   ; his represented as the lambda variable y
c. ∃!x (and (boy x)
            x, lambda(y)[(∃! z) (and (= (dog-of v) z) (love y z))]
   ; his represented as an unbound variable v
```

The context assignment function can assign the null verb phrase in 103 one of the three verb phrases in 104. Hence, three meanings for the elided sentence are provided (shown in 105).

Example 105

```
The man does too.

a. ∃!m (and (man m)
            m, lambda(y)[(∃! z) (and (= (dog-of x) z) (love y z))])
   ; The pronoun is an unbound x
b. ∃!m (and (man m)
            m, lambda(y)[(∃! z) (and (= (dog-of y) z) (love y z))])
   ; The pronoun is bound by the lambda operator
c. ∃!m (and (boy m)
            m, lambda(y)[(∃! z) (and (= (dog-of v) z) (love y z))])
   ; The pronoun is an unbound v
```

Consider the possible interpretations of each of the meanings of the elided sentence. The sloppy interpretation of the elided sentence is available, as shown in 105b. The other two meanings of the elided sentence contain unbound pronoun variables, which must be assigned values by the context assignment function. Compare the meanings of the trigger sentence in 104a and the elided sentence in 105a. In the trigger sentence, the variable x is bound by the operator associated with the definite subject. When the null verb phrase in the elided sentence is assigned the same verb phrase, the pronoun variable becomes unbound. Hence, it should be possible for the context assignment function to assign this variable any value, including the individual that *the boy* denotes. There is no guarantee, however, that the unbound variable will be assigned this value. Hence, a number of impossible interpretations could be provided for the elided sentence. This is not very attractive.

Now compare the meanings of the trigger sentence in 104c and the elided sentence in 105c. In this case, neither variable is bound, and the context assignment function must assign each variable a value. As we pointed out before, there is an assumption that the free pronoun variables in the trigger and elided sentences must be assigned the same value by the context assignment function. Hence, the elided sentence can receive the strict interpretation (given that the context assignment function can pick the individual denoted by *the boy* as the values of the variable), as well as a host of other interpretations, depending on the context assignment function.

We make two comments about this example. First, in Partee and Bach's model, undesirable readings for elided sentences result when pronoun variables bound by an operator in the trigger sentence become unbound in the meaning of the elided sentence[19]. Our approach does not suffer from this problem. We must indicate the antecedent of a pronoun in the trigger sentence before assigning a meaning to the null verb phrase of the elided sentence. Any unbound variable signals the fact that the verb phrase is not an acceptable candidate for the elided verb phrase, unless the null verb phrase occurs within the same sentence as the trigger. Second, the assumption that the unbound variables must be assigned the same values in the trigger and elided sentences seems similar to the assumption that corresponding pronouns in the trigger and elided sentence must both be free or bound by the same operator. The second assumption is unacceptable to the authors, but the first is fine. Both assumptions require a dependency between the meanings of the trigger and elided sentences. The first assumption, however, can be encoded in the context assignment function, whereas the other must be encoded in the parser which generates the meanings for the sentence, with a direct violation of compositionality. However, both assumptions indicate that the meaning of the elided sentence is limited by the meaning of the trigger sentence. Both assumptions are part of the machinery of our approach. That is, we require that the meaning of the elided sentence can be derived only after antecedents for pronouns are known. This information is added to the meaning of the trigger sentence, and affects the possible interpretations of the elided sentence.

One final difficulty with Partee and Bach's approach concerns pronouns whose antecedents are definite noun phrases that cannot bind the pronoun variables. Consider example 38.

Example 38

Every man$_i$ gave the psychiatrist who cares for (his$_i$ mother)$_j$ her$_j$ diary.

This reading of the sentence is certainly reasonable (i.e., where the antecedent for *her* is *his mother* and the antecedent for *his* is *every man*). However, Partee and Bach's model cannot provide this meaning for the sentence. They represent definites as quantified noun phrases, and quantified noun phrases are subject to the complex noun phrase constraint. Hence, because the pronoun *her* is represented as a variable and because *his mother* is affected by the complex noun phrase constraint, the pronoun cannot be bound by *his mother*. The context assignment function must assign the variable corresponding to *her* some constant value. However, given that the antecedent for *his* is *every man*, *his mother* cannot denote a specific individual. Hence, Partee and Bach's model cannot provide the reading indicated in example 38 without introducing some other type of pronoun reference[20].

Partee and Bach are also unable to handle example 106, an example of verb phrase ellipsis similar to example 38.

[19]Sag [45] attempts to fix this problem by defining two different types of variables, pronoun variables and trace variables.

[20]Cooper [13] provides a mechanism for handling donkey sentences (we discuss donkey sentences in Chapter four), which we consider to be similar to example 38.

Example 106

Fred$_i$ gave the psychiatrist who cares for (his$_i$ mother)$_j$ her$_j$ diary.
George$_k$ did too.
Meanings:
1. George gave the psychiatrist who cares for Fred's mother Fred's mother's diary.
2. George gave the psychiatrist who cares for George's mother George's mother's diary.
3. *George gave the psychiatrist who cares for George's mother Fred's mother's diary.
4. *George gave the psychiatrist who cares for Fred's mother George's mother's diary.

Given that *his mother* cannot bind *her*, *her* must be represented as an unbound variable in the trigger sentence. The pronoun *his*, on the other hand, can either be lambda bound, bound by a definite operator, or free. The main problem for Partee and Bach's approach concerns the fact that *her* must be represented as an unbound variable. There is no way to get the sloppy reading of the elided sentence (i.e., meaning two of 106) because of the fact that unbound variables are assigned constant values by the context assignment function. However, their approach does provide the following implausible reading for the elided sentence, *George gave the psychiatrist who cares for George's mother Fred's mother's diary.* In Chapter three, we will demonstrate how our approach handles examples 38 and 106. However, we must first introduce a general representation for definite noun phrases.

Partee and Bach represent pronouns as variables in the meaning of a sentence. Those variables are either bound by some operator in the meaning of the sentence or assigned some value by the context assignment function. Hence, they assume that pronouns act either as bound variables or as constants. They do not introduce a level of logical form to account for verb phrase ellipsis, though their approach suffers from difficulties similar to other approaches that do. In particular, they cannot account for examples where pronoun variables cannot possibly be bound by a definite operator (because of the complex noun phrase constraint) or be replaced by a constant value (because its antecedent is non-static).

2.4.5 Summary of Previous Approaches

We have demonstrated that there are a number of difficulties for past representations of pronouns. In particular, the assumption that pronouns act either as constants or as bound variables causes problems. There is nothing implicit in the meaning of a pronoun that requires pronouns to act as constants or variables. Certainly pronouns often behave like bound variables or constants. However, there seems to be an additional pronoun behavior, that is, pronouns can adopt the functional behavior of their antecedents.

Additionally, we have shown that, other approaches do not obey the computational constraints introduced in Chapter one. Sag's and Webber's approaches violate two or more of these constraints. Reinhart's approach, despite the fact that logical form was not an issue for her, violates our constraints. Finally, Partee and Bach eliminate a level of logical form. However, their approach suffers from the proliferation of impossible interpretations of elided sentences. For this and other reasons, their approach is not a computationally feasible one.

2.5 Summary of Our Approach

In this chapter, we have described a representation of pronouns in logical form. The logical form for a pronoun can be provided using only syntactic and sentence-level information. Additionally, the initial representation can be legally updated to indicate the final meaning of a pronoun. Once the antecedent of a pronoun is determined, the pronoun function is equated with a value depending on the type of its antecedent. For example, when the antecedent of a pronoun is a possessive noun phrase, the pronoun function is equated with the function representing that noun phrase. We also provide a compact way to represent the sloppy identity ambiguity. Hence, our logical form for pronouns satisfies our computational constraints.

We also examined past approaches (discussed in section 2.4). We presented examples that were troublesome for those models. Some of the examples fit nicely into our reduced model of verb phrase ellipsis (i.e., 76 and 80), so we were able to show how our approach handles them. Others, in particular 94, 95, 38 and 106, cannot be discussed until we introduce a general representation for definite noun phrases as functions in the next chapter. These examples are easily handled once we introduce a functional representation for all definite noun phrases. These examples were introduced to demonstrate the fact that pronouns do not always act like bound variables or constants. Instead, we claim that all pronouns act like functions. Functions subsume both the bound variable (i.e., an identity function on a variable) and referential (i.e., a function with no arguments) meanings of a pronoun.

In the next chapter, we provide a general representation for definite noun phrases in logical form. Once we provide this representation, we discuss how it is used to handle examples 94, 95, 38 and 106.

Chapter 3

Definite Noun Phrases

3.1 Introduction

In this chapter, we discuss the representation of singular definite noun phrases in logical form. We concentrate on singular definite noun phrases because plural definites have an additional level of complexity. We motivate this representation using the constraints on logical form proposed in Chapter one, as well as linguistic evidence on the nature of definite noun phrases. First, we discuss the behavior of singular definites in language and then present our representation taking into account the linguistic evidence as well as our constraints on logical form. Finally, we discuss previous representations of definite noun phrases.

3.2 Definite Noun Phrases: Linguistic Evidence

In Chapter two, we showed that when a definite noun phrase is the antecedent for a pronoun, the pronoun cannot always be characterized by the bound-referential dichotomy. We demonstrated that when a pronoun refers intrasententially to a definite noun phrase, the behavior of that noun phrase can be modeled by equating the pronoun with the function used to represent the definite[1]. However, in Chapter two, we did not provide a general representation for all types of definites. Instead, we concentrated on proper nouns and possessive noun phrases. In this section, we characterize the behavior of singular definite noun phrases in English. Then in the next section, we discuss how to represent singular definite noun phrases as functions in order to capture the ways they behave.

Like pronouns, definite noun phrases can be anaphoric. An anaphoric definite can refer to some linguistic entity in discourse, or it can refer to some salient individual in the environment of the speaker/hearer (also known as deixis). The antecedents for anaphoric definites can be noun phrases in previous sentences. Consider example 107.

[1] A pronoun whose antecedent is a definite in subject position is ambiguous. It can either refer to the subject indirectly (hence, we equate the pronoun function with the lambda variable) or directly (hence, we equate the pronoun function with the function representing the definite noun phrase).

Example 107

Fred$_i$ saw his$_i$ cat.
The cat was chasing a mouse.

In 107, the antecedent for *the cat* is *his cat*. Because the antecedent is found in the previous sentence, *the cat* could be represented by the discourse entity assigned to *his cat*. The antecedent for an anaphoric definite can also be some noun phrase within the same sentence. For example, consider 108.

Example 108

Every boy$_i$ who saw (his$_i$ cat)$_j$ take his$_i$ ring chased the animal$_j$.

In this example, *the animal* cannot be replaced by a discourse entity (since *his cat* cannot be assigned some fixed value when the antecedent for *his* is *every boy*). However, the function representing *the animal* could be equated with the function representing *his cat*.

Definites, unlike pronouns, often have a complex syntactic structure. Definites with a complex syntactic structure tend not to be used anaphorically (in contrast to pronouns, which always depend on some antecedent for their meanings). Pronouns and other noun phrases can be embedded in a definite noun phrase, and embedded noun phrases often affect the meaning of a definite noun phrase.

First, consider how embedded pronouns affect the meaning of a definite. Consider example 109.

Example 109

Every boy loves his mother.

The meaning of the definite noun phrase *his mother* depends on the value assigned to the pronoun *his*. If *his* refers to someone in a previous sentence, then *his mother* could be represented as a constant. In contrast, if the antecedent for *his* is *every boy*, then the universal affects the denotation of the definite. In particular, *his mother* changes what it denotes depending on the instantiation of the universal variable. In other words, because of the anaphoric link between *his* and *every boy*, the universal quantifier distributes over the definite. This prevents the definite from acting like a constant. A similar effect is observed in verb phrase ellipsis. Consider example 110.

Example 110

Fred saw his daughter.
George did too.

If *his* refers to some individual outside of the trigger sentence, then *his daughter* must denote the same person in both the trigger and elided sentences. Hence, *his daughter* can be characterized as a constant. However, if the pronoun refers to the subject indirectly, then *his daughter* should denote different individuals in the elided and trigger sentences. Any pronouns within a definite noun phrase can affect the meaning of that noun phrase, regardless of how deeply they are embedded in that noun phrase. Consider example 111. In this example, a pronoun is embedded in a relative clause of a definite noun phrase.

Example 111

```
Every boy loves the girl who loves him.
```

If the antecedent for *him* is *every boy*, then the universal must distribute over *the girl who loves him*, preventing it from behaving like a constant.

On the other hand, simple definite noun phrases (i.e., noun phrases without embedded noun phrases or pronouns) act differently than definite noun phrases with embedded pronouns (e.g., 109, 110, and 111). Consider the following example:

Example 112

```
Every boy loves the girl.
```

In this case, each boy must love the same girl (in contrast to example 109). In verb phrase ellipsis, simple definites must denote the same individual in the trigger and elided sentences, as demonstrated in example 113.

Example 113

```
Fred saw the girl.
George did too.
Meaning:
George saw the same girl as Fred saw.
```

In this case, George must see the same girl as Fred saw (in contrast to example 110). Since there are no embedded pronouns in *the girl*, it must act like a constant across the two sentences.

The meaning of a definite noun phrase can also be affected by embedded noun phrases, particularly those represented as quantified terms. First, consider how a possessive universal noun phrase phrase affects a definite noun phrase's meaning:

Example 114

```
George loves every man's wife.
```

The universal must distribute over the definite noun phrase, so *every man's wife* cannot be described as a constant. Possessive quantified noun phrases always have scope over the matrix noun phrase (as noted by Roberts [41]). Consider another example:

Example 115

```
George redesigned the kitchen in every house.
```

In this example, the definite noun phrase *the kitchen in every house* cannot be described as a constant. However, this intuition is guided by the fact that it is impossible for one kitchen to be in different houses. Hence, our decision about the meaning of the definite in 115 is affected by pragmatics. Actually, a quantified noun phrase contained in a prepositional phrase attached to a definite noun phrase may distribute over the noun phrase, though not necessarily. This ambiguity is discussed by May [34] and Roberts [41]. Consider example 116.

Example 116

`The head of every public authority in New York is rich.`

The head of every public authority in New York is ambiguous. If the universal distributes over the definite, the definite's denotation depends on which public authority we consider. On the other hand, if the universal does not distribute over the definite, then there is one particular person who heads all of the public authorities, and this person is also rich. This ambiguity must be handled when we design our initial representation for a definite noun phrase[2].

Not all quantified noun phrases embedded in a definite noun phrase can distribute over it, however. When quantified noun phrases are contained in a relative clause attached to a definite noun phrase, they cannot distribute over that noun phrase (as noted by Ross [42]). The complex noun phrase constraint prevents the universal noun phrase from moving out of the relative clause to distribute over the definite noun phrase. Consider, for instance, example 117.

Example 117

`George respects the mother who cares for every boy.`

The definite noun phrase *the mother who cares for every boy* seems to denote one specific mother[3].

Next, consider how indefinites embedded in a definite noun phrase affect its meaning. If there are no universal quantifiers in the sentence, the effect of the indefinite is easily missed. Consider example 118.

Example 118

`Fred loves the wife of a colleague.`

On the other hand, when *the wife of a colleague* is placed in a sentence with a universal noun phrase that can have scope over the indefinite, then the effect is more dramatic:

Example 119

`Every man loves the wife of a colleague.`

In this case, *the wife of a colleague* can denote different wives depending on the man under consideration. Whenever a universal noun phrase can have scope over an indefinite embedded in a definite noun phrase, a similar effect can be observed[4]. However, when the indefinite is embedded

[2]Actually, this scoping ambiguity is often difficult to obtain with certain types of prepositions (e.g., *with*). However, I believe that the ambiguity often depends on the meanings of the head nouns, the prepositions, and the objects. This is an interesting research problem, beyond the scope of this thesis.

[3]However, if the definite is a generic definite, it need not denote a single woman. We ignore generic definites in this thesis.

[4]Actually, this phenomenon is strongly affected by which preposition is used. For example:

Example 120
Every man met the boy with a dog.

This sentence seems to mean that every man saw the same boy. However, there is a lot of pragmatics going on here, as the following example shows:

Example 121
Each man redesigned the kitchen in a house.

in a relative clause attached to a definite noun phrase, a universal noun phrase outside of the relative clause cannot have scope over the indefinite. In such a case, the universal has no affect on the meaning of the definite, either. Consider example 122.

Example 122

```
Every man chased the cat that scratched a dog.
```

In this case, the same cat is chased by all of the men.

In the case of verb phrase ellipsis, embedded noun phrases can also affect the meaning of a definite noun phrase. For example, when an indefinite is contained in a prepositional phrase attached to a definite noun phrase (or is the possessive noun phrase in a definite noun phrase), then the definite noun phrase need not denote the same individual in the trigger and elided sentences, as shown in 123.

Example 123

```
Fred loves the wife of a colleague.
George does too.
Meanings:
1. George loves the same woman.
2. George love the wife of a different colleague.
```

However, this effect disappears when the indefinite is embedded in a relative clause. When the indefinite is contained in a relative clause attached to a definite noun phrase, the definite must denote the same individual in the trigger and elided sentences, as demonstrated by example 124.

Example 124

```
Fred loves the woman who chased a cat.
George does too.
Meaning:
George loves the same woman as Fred.
```

When a universal embedded in a definite noun phrase distributes over it, the definite cannot denote one single individual. However, if a universal (without embedded pronouns) distributes over a definite in the verb phrase of a trigger sentence, that definite denotes the same group of individuals in the elided sentence as in the trigger sentence, as shown in 125.

Example 125

```
Fred loves every man's wife.
George does too.
Meaning:
George loves every man's wife.
```

In this case, George and Fred love the same group of wives. Despite the fact that *every man* distributes over the definite noun phrase, the definite must denote the same group of individuals

---

In this case, each of the men can redesign a different kitchen.

in the trigger and elided sentences. On the other hand, suppose the restriction of the universal contains a pronoun whose antecedent is the subject of the trigger sentence. For example:

Example 126

Fred$_i$ loves the wife of every man he$_i$ knows.
George$_j$ does too.
Meanings:
1. George loves the wife of every man Fred knows.
2. George loves the wife of every man George knows.

Because the antecedent of *he* is the subject, the group that the universal noun phrase denotes can change, depending on the meaning chosen for the pronoun *he*. Hence, the definite noun phrase can denote a different group in the elided sentence than in the trigger sentence.

Thus, we have shown that the meaning of a definite noun phrase is affected by three factors: whether it is used anaphorically, whether it contains pronouns, and whether it contains quantified noun phrases not subject to the complex noun phrase constraint. If used anaphorically, it should behave in a way consistent with its antecedent, just like a pronoun. If it contains pronouns, then its meaning should depend on the antecedents chosen for those pronouns. If it contains embedded quantified noun phrases (not subject to the complex noun phrase constraint), then those embedded noun phrases may affect the meaning of the definite noun phrase. Our logical form representation for a definite noun phrase must be compatible with these behaviors, otherwise the representation will violate our computational constraints. In the next section, we describe how to represent definites as functions in logical form. We demonstrate how this representation is used to handle examples of verb phrase ellipsis as well as the examples that were problematic for the approaches discussed in Chapter two.

3.3 Our Representation

3.3.1 Introduction

In this section, we develop a representation for definite noun phrases as functions in logical form. As we will show, this representation allows us to model both anaphoric definites and definites whose meanings are affected by embedded pronouns and quantified noun phrases. Also, our initial representation for definite noun phrases is consistent with the computational constraints from Chapter one. The reader should keep in mind at the start that the logical form for a definite noun phrase is a composite representation for all of its possible meanings, given its structure and position in a sentence. To be consistent with our constraints, neither antecedents nor quantifier scoping information can be known before the initial representation of a definite noun phrase is provided. Hence, it should not be surprising that initially definite functions have no single model-theoretic interpretation. Once antecedents for all embedded anaphoric noun phrases are determined and quantifier scoping has been stipulated, a single (possibly model-theoretic) interpretation for the definite function can be provided.

In the next section, we propose a representation for definite noun phrases which allows us to model anaphoric definites and definites affected by embedded pronouns and quantified noun phrases. We also provide a way to further specify this representation once quantifier scoping and antecedents for anaphoric noun phrases are determined. After this, we demonstrate how our representation is used

to handle verb phrase ellipsis, the problematic examples from Chapter two, and some historical definite examples.

3.3.2 The Initial Representation of Definite Noun Phrases

In this section, we introduce a general logical form representation for singular definite noun phrases. This representation must handle both the non-anaphoric distributive readings of a definite noun phrase and possible anaphoric readings. To handle the anaphoric aspect of definite noun phrases, we borrow the functional representation used for pronouns. However, the functional representation for definites is an adaptation of the pronoun function. Definite noun phrases are not always anaphoric and are often affected by embedded pronouns and quantified noun phrases (as we have already shown in the previous section).

The logical form representation for a definite noun phrase presents a challenge to our approach. To be consistent with the modularity constraint, we must develop a logical form representation for a definite which can be generated before we know its antecedent (if it is anaphoric) or the antecedents for any embedded pronouns. To obey the compactness and formal consistency constraints, a definite's initial representation must be consistent with all the ways it can possibly act. When more information is available about the meaning of a definite noun phrase, we must be able to update logical form in a way compatible with its initial representation.

To obey our logical form constraints, we initially represent a definite as a function in logical form. The initial representation is derived using the following specification.

The Initial Representation of a Singular Definite Noun Phrase

> A singular definite noun phrase is initially represented as a function in logical form. Each definite function consists of a unique name (i.e., *def* with a unique integer appended to it), a list of arguments, and some restriction. The argument list of the function consists of all of the variables associated with lambda operators that have scope over its position, any variables corresponding to non-subject quantified noun phrases that could bind a pronoun in that position, and any variables associated with embedded quantified noun phrases that are not subject to the complex noun phrase constraint[5]. The restriction of a definite function is determined by parsing the rest of the noun phrase following the definite article. Any quantifiers associated with non-possessive quantified noun phrases are initially placed inside the restriction. In contrast, possessive quantifiers are placed outside of the definite function since they must always have scope over it.

Notice that definite functions differ from pronoun functions in three obvious ways. Definite functions are named differently than pronoun functions, and have restrictions. Also, definite functions can have arguments introduced by embedded quantified noun phrases. In contrast, pronouns do not contain embedded noun phrases. However, like pronoun functions, definite functions are provided using only syntactic and sentence-level information (consistent with the modularity constraint). Because a definite noun phrase is represented as a function of all of the quantified variables that can distribute over it (because of pronoun resolution or because embedded quantified noun phrases could distribute over the definite function), the initial representation of the definite is consistent

[5] In particular, quantified variables corresponding to quantified noun phrases contained in a relative clause attached to a definite should not be included in the argument list of the definite function since they never distribute over it.

with the range of behaviors that noun phrase can have. A definite function is a composite representation for a definite noun phrase based on its structure and position in a sentence (the semantics for our logical form is given in Appendix B). This initial representation is compatible with both its anaphoric side and its distributive side. In fact, this composite representation can be thought of as an upper bound on the behavior of a definite, where the lower bound is a constant. Later when more information is available, we tighten the bounds on the definite function[6].

Now that the initial representation for a definite has been specified, consider some examples demonstrating this representation. First, consider example 127.

Example 127

```
Every man showed every boy his picture.
```

The initial representation of *his picture* is shown in 128.

Example 128

```
Every man showed every boy his picture.

∀x: (man x)
 x, λ(y)(show y ((def1 y z) | (and (picture (def1 y z))
                                   (possess (his2 y z) (def1 y z))))
            [∀z: (boy z) z])
```

Notice that *his picture* is represented as a function called def_1. Because *every man* is the subject of the sentence and it c-commands *his picture*, the argument list of the definite function must include the lambda variable *y*. Also because *every boy* c-commands *his picture*, the variable *z* is also included in the argument list. The restriction of the function is the conjunction of statements following the vertical bar. The bar separating the function from its restriction is syntactic sugar and is expanded in much the same way as an existential's restriction[7], as shown in 129.

Example 129

```
Every man showed every boy his picture.

∀x: (man x)  x, λ(y)(and (show y (def1 y z) [∀z: (boy z) z])
                         (picture (def1 y z))
                         (possess (his2 y z) (def1 y z)))
```

To be consistent with our logical form constraints, the initial representation of *his picture* (shown in 128) is provided before the antecedent of *his* is known. Additionally, the representation of the definite must be compatible with any behavior the definite can adopt once the antecedent for the pronoun *his* is determined. The argument list of the definite function contains all of the variables that can affect the meaning of its embedded pronoun. Because the antecedent of the pronoun *his* can be *every boy*, *every man*, or some noun phrase in a previous sentence, *his* is initially represented

[6]One might also think of pronoun and definite functions as variables of type function. The values assigned to these variables must be functions whose argument lists are subsets of the composite's argument list.

[7]However, this expansion should not be carried out until it is possible to settle on the final meaning of the definite noun phrase. There is good reason for keeping the restriction with the function, as we will show in the next section.

as a function of the lambda variable *y* (associated with the subject *every man*) and the universal variable *z* (associated with *every boy*). Because the argument list of the pronoun function includes the variables *y* and *z*, the definite function representing *his picture* is also represented as a function of those arguments. However, because the function representing *his picture* (i.e., (def_1 y z)) is a composite representation, it must be limited once we determine the antecedent for the embedded pronoun (as we show in the next section).

Our initial representation for definite noun phrases can also be used in verb phrase ellipsis. Consider example 130.

Example 130

```
Fred chased his dog.
George did too.
Meanings:
1. George chased some third party's dog (and the same one as Fred).
2. George chased Fred's dog.
3. George chased George's dog.
```

A composite representation for *his dog* (contained in the trigger verb phrase in 130) must be compatible with the possible meanings of the pronoun *his*. To handle potential sloppy readings, the argument list of a definite function must include the lambda variables that an embedded pronoun function can be equated with. Consider the initial representation of the trigger sentence from example 130.

Example 131

```
Fred chased his dog.

((def1) | (name (def1) Fred)),
   λ(x)(chase x ((def2 x) | (and (dog (def2 x))
                                 (possess (his3 x) (def2 x)))))
```

Because the subject *Fred* can be the antecedent for *his*, the pronoun is initially represented as a function of the lambda variable *x*. We cannot determine the antecedent for *his* before providing the representation for the definite noun phrase. Hence, the definite is also initially represented as a function of the lambda variable. However, because (def_2 x) is a composite representation for *his dog*, it must be limited once we determine the antecedent for the embedded pronoun. As we will show in section 3.3.4, a definite's composite meaning must be restricted to provide the possible strict reading of a definite noun phrase in verb phrase ellipsis.

When we provide the initial representation for a definite noun phrase, we also account for embedded quantified noun phrases not subject to the complex noun phrase constraint. Consider the following example:

Example 132

```
Every man's wife loves him.
```

The noun phrase *every man's wife* does not denote a single wife; *every man* distributes over the definite noun phrase. Because a possessive quantified noun phrase embedded in a definite noun phrase must always distribute over the definite (as discussed in section 3.2), the variable associated

with the quantifier is included in the argument list of the definite function and the quantifier is pulled out of the restriction of the definite function to distribute over it. Furthermore, that quantified noun phrase can have scope over other noun phrases in the sentence (particularly those noun phrases the matrix noun phrase can have scope over). The initial representation for *every man's wife* is shown in example 133.

Example 133

```
Every man's wife loves him.

∀x: (man x) ((def1 x) | (and (wife (def1 x))
                             (possess x (def1 x)))), λ(y)(love y (him2 x y))
```

Notice that the definite function def_1 is a function of the universal variable x. Also notice that the universal operator associated with *every man's* distributes over the definite function. Finally, notice that *him* is represented as a function of x and y.

On the other hand, quantified noun phrases contained in relative clauses attached to a definite noun phrase cannot distribute over the definite noun phrase. The complex noun phrase constraint prevents a quantifier from moving out of a relative clause to distribute over the definite. In such a situation, the definite acts like a simple definite noun phrase (assuming the restriction contains no embedded pronouns). Consider the following example:

Example 134

```
The child who cares for every man is happy.
```

The initial representation for *the child who cares for every man* is shown in 135.

Example 135

```
The child who cares for every man is happy.

((def1) | (and (child (def1))
               ((def1), λ(x)(care x (for [∀y: (man y) y])))))), λ(z)(happy z)
```

Notice that *the child who cares for every man* is represented as a function with no arguments, and thus acts like a constant. Additionally, the universal quantifier cannot distribute over the definite in this case (since its variable is not included in the argument list of the function). Notice that we do not provide an explicit representation for *who* in this thesis. Instead, we simply represent it by using the relative head's representation. In 135, *who* is represented as a definite function[8].

Universal noun phrases embedded in a relative clause attached to a definite noun phrase cannot bind a pronoun outside the relative clause, as noted by May [34] and Roberts [41]. Consider example 136.

Example 136

```
*The child who cares for every man_i visits him_i.
```

[8] If the relative head is quantified, we represent the relative pronoun using the variable associated with the quantifier. There are many good reasons for providing an explicit representation for relative pronouns. However, this is an issue beyond the scope of this work.

The universal noun phrase *every man* cannot bind the pronoun *him*. When quantifiers subject to the complex noun phrase constraint cannot distribute over a definite function, they cannot bind a pronoun in the matrix sentence (i.e., they cannot c-command that pronoun, hence they cannot bind it). Consider the initial representation of the sentence in 136.

Example 137

```
The child who cares for every man visits him.

((def1) | (and (child (def1))
               ((def1), λ(x)(care x (for [∀y: (man y) y])))),
  λ(z)(visit z (him2 z))
```

The pronoun *him* is represented as a function of the lambda variable *z*. Because the universal noun phrase does not c-command the pronoun *him*, the pronoun function's argument list does not contain the variable *y*. Hence, our initial representation of the pronoun *him* cannot be replaced by the universal variable of *every man* (without violating formal consistency).

If a quantified noun phrase is contained in a prepositional phrase attached to a definite noun phrase, the representation of the definite differs from both the possessive and relative clause cases. Consider example 116.

Example 116

```
The head of every public authority in New York is rich.
```

As discussed in section 3.2, the definite noun phrase in this sentence is ambiguous. One meaning is shown in 138.

Example 138

```
The head of every public authority in New York is rich.

∀x: (and (public-authority x) (in x New York))
  ((def1 x) | (head-of x (def1 x))), λ(y)(rich y)
```

Notice that in this case, the individual denoted by the noun phrase, *the head of every public authority in New York*, depends on the public authority we consider. The second meaning of the definite in 116 is shown in 139.

Example 139

```
The head of every public authority in New York is rich.

((def1) | (head-of [∀x: (and (public-authority x) (in x New York)) x]
                   (def1))), λ(y)(rich y)
```

In this case, there is one particular person who heads all of the public authorities, and this person is also rich. Our initial representation for *the head of every public authority* must be compatible with the definite's two possible meanings. In fact, when a quantified noun phrase is the object of a preposition attached to a definite noun phrase, we include its variable in the argument list of the initial definite function. Additionally, we place the quantifier inside the restriction of the definite function. Hence, the initial representation for *the head of every public authority* is shown in 140.

Example 140

The head of every public authority in New York is rich.

```
((def1 x) | (head-of [∀x: (and (public-authority x) (in x New York)) x]
                     (def1 x))), λ(y)(rich y)
```

Notice that the quantifier is placed inside the restriction of def_1, and the variable x is placed in the argument list (for the semantics of such a function, see Appendix B). Later, when we decide whether or not the quantifier distributes over the definite, the initial representation is limited, as discussed in the next section.

The argument list of a definite function includes the arguments that a pronoun function in the same position would have plus any quantified variables associated with embedded quantified noun phrases not subject to the complex noun phrase constraint. However, someone might propose that a definite function should inherit all of the arguments of embedded pronouns. Example 141 demonstrates that definite functions should not inherit all of the arguments of its embedded pronoun functions.

Example 141

Every man saw the woman who showed every boy his dog.

Consider the initial logical form for this sentence, shown in 142.

Example 142

Every man saw the woman who showed every boy his dog.

```
∀x: (man x)
 x, λ(y)(see y ((def1 y) |
                 (and (woman (def1 y))
                      (def1 y),
                        λ(z)(show z ((def2 y z w) |
                                        (and (dog (def2 y z w))
                                             (possess (his3 y z w)
                                                      (def2 y z w))))
                                    [∀w: (boy w) w])))))
```

The pronoun *his* is represented as a function of two lambda variables (i.e., y and z) and the universal variable w. However, the function representing *the woman who showed every boy his dog* is represented as a function of the variable y (corresponding to *every man*). Despite the fact that *every boy* can be the antecedent for *his*, the argument list for the function representing *every woman who showed every boy his dog* should not include the variable w (because of the complex noun phrase constraint). In other words, pronouns embedded in a definite noun phrase can be bound by quantified noun phrases whose variables should not be included in the argument list of the definite function (as shown in example 142). Hence, definite functions should inherit only those arguments from embedded pronoun functions associated with noun phrases outside of the function.

Because a definite function is a composite representation for all of the possible meanings of a definite noun phrase, we must restrict the function in certain ways once the final meaning of the

noun phrase is available (e.g., before we derive the meaning of an elided sentence from a trigger verb phrase containing a definite function (we discuss this in section 3.3.4)). The initial representation of a definite places an upper and lower bound on the ways that definite noun phrase can behave. The lower bound is a constant, while the upper bound is the composite function. To settle on a final meaning for the definite, we must determine the precise behavior of the definite in a sentence. We provide two methods to pinpoint the behavior of a definite function. They are discussed in the next section.

3.3.3 Limiting Composite Definite Functions

Because a definite function is a composite representation for the possible meanings of a definite noun phrase based on its structure and location in a sentence, it is an excellent initial representation for a definite. However, because it is a composite representation, it must be limited before a final interpretation for a sentence is possible. Definite functions can be limited in two ways. If the definite is used anaphorically, we equate the definite function with some value consistent with its antecedent (just like a pronoun function). Otherwise, we apply a constraint that limits the argument list of the function to include only necessary variables.

If a definite noun phrase is used anaphorically, then the definite function should be equated with some value depending on the antecedent's type. For example, if the antecedent of a definite noun phrase occurs in another sentence, we equate the definite function with a discourse entity. Consider example 143.

Example 143

```
Fred loves a woman.
The woman loves him.
```

The initial representations for these two sentences are shown in 144.

Example 144

```
Fred loves a woman.

((def1) | (name (def1) Fred)), λ(x)(love x [∃y: (woman y) y])

The woman loves him.

((def2) | (woman (def2))), λ(z)(loves z (him3 z))
```

Given that the antecedent of *the woman* is *a woman*, then the function representing *the woman* must be equated with the discourse entity that *a woman* evokes (following Webber [49]). This is shown in 145.

Example 145

```
The woman loves him.

(and ((def2) | (woman (def2))), λ(z)(loves z (him3))
     (= (def2) woman22))
```

When the antecedent occurs in a different sentence than the anaphoric definite (as in 145), the anaphora resolution module must ensure that the antecedent's discourse entity is compatible (in number, as well as a host of other features) with the definite noun phrase. If they are incompatible, then the update is not allowed. For example, consider the sentences in example 146.

Example 146

```
Fred visits everyone's mother.
Everyone loves the woman.
```

Everyone's mother would be represented as a function of the universal variable corresponding to *everyone*. Using Webber's [49] method for constructing discourse entities, a discourse entity denoting a set of mothers would be created for *everyone's mother*. Because *the woman* is anaphoric, the anaphora resolution module would search for that noun phrase's antecedent. However, because *the woman* is incompatible with the discourse entity created for *everyone's mother*, *everyone's mother* cannot be the antecedent.

Antecedents for definite noun phrases can also occur within the same sentence. The anaphora resolution module must check possible intrasentential antecedents for compatibility with the definite. In particular, quantified antecedents for definite noun phrases are typically embedded in a noun phrase that c-commands the definite. Consider the following example:

Example 147

(The owner of every dog_j)$_i$ is afraid of the animal_j.

The initial representation of this sentence is shown in 148.

Example 148

```
The owner of every dog is afraid of the animal.

((def1 x) | (and (owner (def1 x))
                 (of (def1 x)  [∀x: (dog x) x]))),
   λ(y)(afraid-of y ((def2 x y) | (animal (def2 x y))))
```

Because *every dog* is embedded in *the owner of every dog* and it is not affected by the complex noun phrase constraint, the anaphora resolution allows the universal to be the antecedent for *the animal*. However, to obey the formal consistency constraint, the function representing the anaphoric definite must be compatible with the representation of its antecedent. Because the argument list of the function representing *the animal* contains the universal variable x, *every dog* is compatible with *the animal*. Hence, (def_2 x y) can be equated with the universal variable x (as shown in 149).

Example 149

(The owner of every dog_i)$_j$ is afraid of the animal_i

```
((def1 x) | (and (owner (def1 x))
                 (of (def1 x)  [∀x: (dog x) x]))),
   λ(y)(and (afraid-of y ((def2 x y) | (animal (def2 x y))))
            (= (def2 x y) x))
```

A definite noun phrase can also be the antecedent for an anaphoric definite. If a definite noun phrase is embedded in another noun phrase, then the anaphora resolution module might propose it as the antecedent for a definite anaphor. Then, if the anaphor's argument list is compatible with the antecedent's argument list[9], our approach allows the anaphoric link between two definite noun phrases. Consider the following example:

Example 150

Every man$_i$ told ((his$_i$ mother's)$_j$ psychiatrist)$_k$ about the old lady's$_j$ diary.

Consider the initial representation of this sentence, shown in 151.

Example 151

Every man told his mother's psychiatrist about the old lady's diary.

```
∀x: (man x)
 x, λ(y)(tell y ((def1 y) |
                  (and (psychiatrist (def1 y))
                       (possess ((def2 y) | (and (mother (def2 y))
                                                 (possess (his3 y)
                                                          (def1 y))))
                                (def1 y))))
              (about ((def4 y) | (and (diary (def4 y))
                                      (possess ((def5 y) |
                                                (old-lady (def5 y)))
                                               (def4 y))))))
```

Assuming that *the old lady* is an anaphoric definite, the anaphora resolution module must provide antecedents for *his* and *the old lady*. Suppose both of the anaphoric noun phrases have intrasentential antecedents. Also, assume that the antecedent for *his* is *every man*. The anaphora resolution module allows *his mother* to be the antecedent for *the old lady* (because the antecedent is embedded in another noun). Additionally, because the argument lists of the two functions are compatible, we can add this information to the logical form. This update is shown in 152.

[9]The argument list of the antecedent's function must be a subset of (or it must be possible to limit it to be a subset of) the arguments of the anaphoric definite.

Example 152

Every man told his mother's$_i$ psychiatrist about the old lady's$_i$ diary.

```
∀x: (man x)
 x, λ(y)(tell y ((def1 y) |
                  (and (psychiatrist (def1 y))
                       (possess ((def2 y) | (and (mother (def2 y))
                                                 (possess (his3 y) (def1 y))
                                                 (or (= (his3 y) y)
                                                     (= (his3 y) x))))
                                (def1 y))))
            (about ((def4 y) | (and (diary (def4 y))
                                    (possess ((def5 y) |
                                              (old-lady (def5 y)))
                                             (def4 y))
                                    (= (def5 y) (def2 y))))))
```

Because the function representing ***his mother*** is compatible with the function representing ***the old lady***, we equate (or replace) (def_5 y) with (def_2 y) to provide a reasonable meaning for ***the old lady***. Hence, when a definite is used anaphorically and its antecedent is located and checked for compatibility, we replace the definite function with some value corresponding with its antecedent.

We must also limit the meanings of non-anaphoric definite functions. Before limiting the function, however, we must determine antecedents for embedded anaphoric noun phrases and decide whether or not the quantifiers of embedded quantified noun phrases (not subject to the complex noun phrase constraint) should distribute over the definite. Consider the initial representation of the sentence in example 128.

Example 128

Every man showed every boy his picture.

```
∀x: (man x)
 x, λ(y)(show y ((def1 y z) | (and (picture (def1 y z))
                                   (possess (his2 y z) (def1 y z))))
            [∀z: (boy z) z])
```

The definite function def_1 is a function of all of the variables that can affect its meaning. However, once we determine the antecedent for the embedded pronoun, the argument list of the function should be limited so that only the intended meaning of the function is possible. In fact, we must limit the argument list of the function to include only those variables associated with operators that distribute over the definite noun phrase directly, or distribute over some other noun phrase that an embedded pronoun is referentially dependent on. We propose a constraint which hinges on the idea that the argument list of a definite function and its restriction mutually constrain each other. If a certain variable is not contained in a definite function's argument list, then that variable cannot affect what the function denotes (and that variable cannot be free within the restriction of the function). However, if a variable is contained in the function's argument list, and that variable is free in the function's restriction (not including the function's argument list), then that variable affects the function. Additionally, if the variable is bound in the restriction of the definite function, then that variable should be eliminated from the argument list for that function (to avoid leaving an unbound variable in the function's argument list). Using these observations, we propose the

argument reduction constraint[10] to limit a definite function once additional information is available about that noun phrase. In particular, we must determine all antecedents for embedded anaphoric noun phrases and decide whether quantifiers contained in the restriction of the definite should distribute over the definite function. Once this information is available, we replace the original function by a function over only the necessary arguments, limiting the composite representation to its final meaning.

The Argument Reduction Constraint:

> To provide the intended meaning for a non-anaphoric definite noun phrase, we limit the argument list of the definite function to include only necessary arguments. If we replace each embedded anaphoric function (pronoun or definite) with its value and apply the argument reduction constraint to any embedded non-anaphoric definite functions, then we can determine the necessary arguments of the function. These include all variables not bound by operators in the function's restriction plus all of the necessary arguments of reduced (and embedded) definite functions not bound by operators in the definite function's restriction. Any other arguments are superfluous, and should be omitted. Once the necessary arguments have been determined, the original definite function is replaced by a new function over only the necessary arguments, using equality.

By using this constraint we limit a definite noun phrase's initial composite representation to its final meaning. As we will show, this step is necessary to ensure that the function representing the final meaning of a definite is well-formed and that its argument list contains only those arguments that affect its meaning. One interesting case occurs when the restriction on a definite function contains no free variables and all of the functions mentioned in the restriction also act like constants. In this case, the definite function is replaced by a function with no arguments (and hence, should act like a constant).

Consider how we limit the function, (def_1 y z), from example 128 following pronoun resolution.

Example 128

Every man showed every boy his picture.

```
∀x: (man x)
 x, λ(y)(show y ((def1 y z) | (and (picture (def1 y z))
                                   (possess (his2 y z) (def1 y z))))
          [∀z: (boy z) z])
```

If we decide that *every boy* is the antecedent for *his*, then we update the logical form, as shown in 153.

[10] As the reader will see, this constraint has been influenced by the ways a definite description provides distributive readings. In particular, a definite description will provide a distributive reading when it contains a variable bound by operators outside of the description.

Example 153

```
Every man showed every boy_i his_i picture.

∀x: (man x)
 x, λ(y)(show y ((def_1 y z) | (and (picture (def_1 y z))
                                    (possess (his_2 y z) (def_1 y z))
                                    (= (his_2 y z) z)))
            [∀z: (boy z) z])
```

Consider the restriction of (def_1 y z). Because the pronoun function (his_2 y z) is replaced by the variable *z*, *z* is the only variable free in the restriction of (def_1 y z) (if we ignore the function itself). Hence, the argument reduction constraint replaces the function (def_1 y z) with some new function over *z*, as shown in 154.

Example 154

```
Every man showed every boy_i his_i picture.

∀x: (man x)
 x, λ(y)(and (show y ((def_1 y z) | (and (picture (def_1 y z))
                                         (possess (his_2 y z) (def_1 y z))
                                         (= (his_2 y z) z)))
                    [∀z: (boy z) z])
             (= (def_1 y z) (def_3 z)))
```

Because equality means that the leftmost function should be replaced by the right-most value (which is either a variable or a function) and because of the meaning of the vertical bar in the restriction of the function, this representation can be simplified[11], as shown in 155.

Example 155

```
Every man showed every boy_i his_i picture.

∀x: (man x) x, λ(y)(and (show y (def_3 z) [∀z: (boy z) z])
                        (picture (def_3 z))
                        (possess z (def_3 z)))
```

Our approach also provides reasonable final meanings of ***his picture*** given other possible antecedents of ***his***.

Now consider how we would update the initial representation of the sentence in 140.

Example 140

```
The head of every public authority in New York is rich.

((def_1 x) | (head-of [∀x: (and (public-authority x) (in x New York)) x]
                      (def_1 x))), λ(y)(rich y)
```

If we decide *every public authority in New York* distributes over the definite function, then the

[11] We introduce simplification to show the meanings of the updated logical forms without a lot of distracting equality statements. However, we often omit this step because of space concerns.

universal quantifier ∀x is moved out of the restriction prior to applying the argument reduction constraint, as shown in 156.

Example 156

```
The head of every public authority in New York is rich.

∀x: (and (public-authority x) (in x New York))
  ((def1 x) | (head-of x (def1 x))), λ(y)(rich y)
```

Because the variable x is free in the restriction of (def_1 x), the function must retain the variable in its argument list. On the other hand, if ***every public authority in New York*** does not distribute over the function, then the quantifier remains in the restriction (as shown in 157).

Example 157

```
The head of every public authority in New York is rich.

((def1 x) | (head-of [∀x: (and (public-authority x) (in x New York)) x]
                     (def1 x))), λ(y)(rich y)
```

Because the restriction of (def_1 x) contains no free variables, the argument reduction constraint replaces it with a function with no arguments, as shown in 158.

Example 158

```
The head of every public authority in New York is rich.

(and ((def1 x) | (head-of [∀x: (and (public-authority x) (in x New York)) x]
                          (def1 x))), λ(y)(rich y)
     (= (def1 x) (def2)))
```

Hence, our approach provides both of the possible meanings for the definite noun phrase in example 116 without using multiple initial representations (which would violate compactness).

Now that we have introduced the initial representation for a definite noun phrase and a way to limit this representation once more information is available, we briefly discuss two possible problems for the approach. The first difficulty concerns the fact that quantified objects of prepositions attached to a definite noun phrase can bind matrix pronouns only when they distribute over the definite (observed by Roberts [41][12] and May [34]). Consider example 159.

Example 159

```
The secretary of every spy keeps an eye on him.
```

In this case, the noun phrase ***every spy*** can bind the pronoun ***him*** only when it has scope over the definite noun phrase, giving the definite noun phrase a distributive reading. Does this make our representation of the pronoun ***him*** contingent on a quantifier scoping decision? The answer is a

[12]Roberts [41] modifies the definition of c-command to handle examples like 159. Her rule states that if a prepositional-phrase-attached quantified noun phrase distributes over its container, then it c-commands the same noun phrases as the containing noun phrase. Because the quantifier corresponding to *every spy* in 159 does not necessarily distribute over the definite, it optionally c-commands *him*.

guarded no.

Recall that we represent a pronoun as a function of all of the variables corresponding to quantified noun phrases that can potentially bind it. Hence, our initial representation for the sentence in 159 is shown in 160.

Example 160

```
The secretary of every spy keeps an eye on him.

((def1 x) | (and (secretary (def1 x))
                 (of (def1 x)
                     [∀x: (spy x) x])))
   λ(y)(keep y [∃z: (eye z) z] (on (him3 x y z)))
```

Because *every spy* can potentially bind *him*, the argument list of him_3 must contain the variable x. However, because the quantifier is unable to bind the pronoun unless it distributes over the definite, we must make certain that the pronoun function is not equated with the variable x, unless its quantifier distributes over the definite function. This requirement could certainly be incorporated into a pronoun resolution module or it could be formulated as a well-formedness constraint on logical form.

Attachment ambiguity poses a problem for our approach. Consider the following example:

Example 161

```
Every man saw the boy with his binoculars.
```

This sentence has two potential parses, as shown in Figure 3.1. Notice that in the first parse tree (shown in Figure 3.1a), the prepositional phrase is attached to the noun phrase, and in the other (shown in Figure 3.1b), it is attached to the verb phrase. These two parse trees give different logical forms. The logical form for the first parse tree is shown in 162.

Example 162

```
Every man saw the boy with his binoculars.

∀x: (man x)
 x, λ(y)(saw y ((def1 y) |
                 (and (boy (def1 y))
                      (with (def1 y)
                            ((def2 y) | (and (binoculars (def2 y))
                                             (possess (his3 x)
                                                      (def2 y))))))))
```

In contrast, when the prepositional phrase is attached to the verb phrase, the logical form for the sentence is shown in 163.

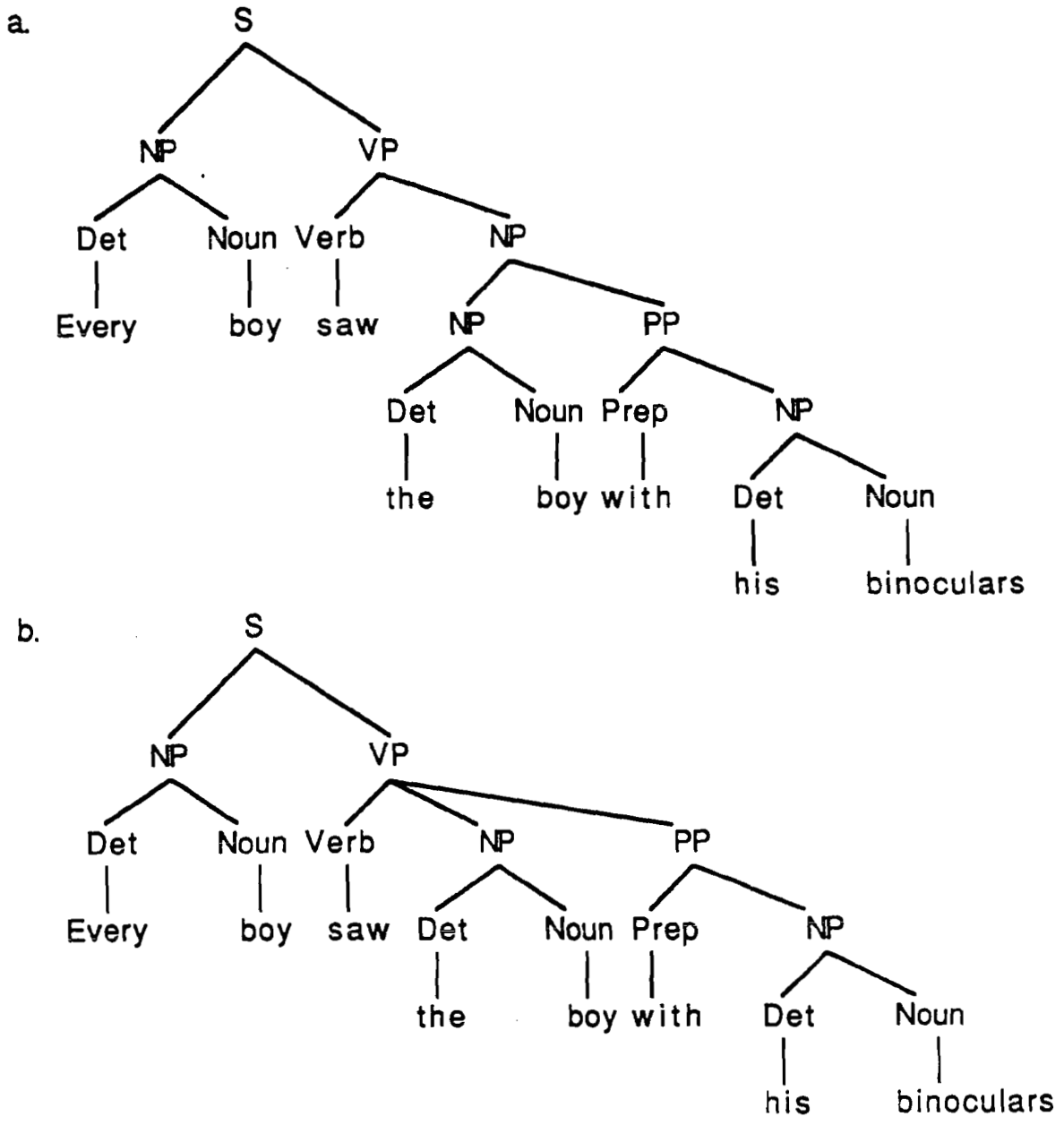

Figure 3.1: Parse Trees for Example 161

Example 163

Every man saw the boy with his binoculars.

```
∀x: (man x)
 x, λ(y)(saw y ((def1 y) | (boy (def1 y)))
             (with ((def2 y) | (and (binoculars (def2 y))
                                    (possess (his3 x) (def2 y))))))
```

These two logical forms are quite different. Hence, attachment ambiguity must be resolved before we generate the logical form for a sentence (violating the modularity constraint) or we must be prepared to provide multiple logical forms for a single sentence (violating the compactness constraint). However, because it is impossible to provide the representation for a noun phrase until we know what is contained in that noun phrase, this violation of our computational constraints is impossible to circumvent[13].

Now that we have introduced our approach for handling definite noun phrases, we are obligated to see how it fares with respect to the logical form constraints introduced in Chapter one. Compactness is observed, once we resolve attachment ambiguities. It would be nice to have a way to represent a definite noun phrase without deciding on attachment ambiguities, but it is difficult to represent something before we know what is included in it. However, because we can determine a representation for a definite noun phrase which captures both its anaphoric role and its role as a structurally complex noun phrase (once attachment ambiguities are resolved), we have largely met the demands of the compactness constraint. We have also specified how to represent a definite noun phrase using only syntactic and sentence-level information, in keeping with the modularity constraint. And finally, we have shown how to update the logical form representation for a definite noun phrase, in keeping with the formal consistency constraint. Any update to logical form is a specialization of that composite representation. Hence, we have provided a general representation for singular definite noun phrases as functions, and this representation allows us to obey our three computational constraints.

Now that we have introduced our representation of definite noun phrases, we demonstrate how it is used to handle verb phrase ellipsis, some examples from Chapter two, and some historically interesting examples.

3.3.4 Verb Phrase Ellipsis

To handle verb phrase ellipsis, we must limit the meaning of a definite function contained in the verb phrase of a trigger sentence before providing the interpretation of an elided sentence. Unless we constrain the initial representation of a definite in the trigger verb phrase before deriving the meaning of the elided sentence, we cannot provide strict meanings for the elided sentence. To see this, consider the meaning of an elided sentence whose trigger sentence contains a definite noun phrase without embedded pronouns.

[13] In fact, prepositional phrase attachment is a problem for anyone who wants to provide representations of noun phrases.

Example 164

```
Fred chased the cat.
George did too.
Meaning:
George chased the same cat as Fred.
```

Given that the antecedent for *the cat* is not contained in the sentence, it must act like a constant. In other words, *the cat* must denote the same cat in the trigger and elided sentences. Consider the initial representation of the trigger sentence, shown in 165.

Example 165

```
Fred chased the cat.

((def1) | (name (def1) Fred)), λ(x)(chase x ((def2 x) | (cat (def2 x)))
```

Notice that both *Fred* and *the cat* are represented as functions. Since def_1 is a function with no arguments, it acts like a constant. On the other hand, the argument list of the function representing *the cat*, def_2, initially contains the lambda variable x.

Suppose that we derive the meaning of the elided sentence using the representation of the verb phrase shown in 165, without limiting the function representing *the cat*.

Example 166

```
George did too.

((def3) | (name (def3) George)), λ(x)(chase x ((def2 x) | (cat (def2 x)))
; George chased a different cat than Fred chased.
```

The meaning expressed in 166 is not a possible meaning for the elided sentence. Hence, before deriving the meaning of the elided sentence from the representation of its trigger sentence, we must determine the meanings of embedded noun phrases and then apply the argument reduction constraint. Assume that *the cat* is non-anaphoric. Hence, the argument list of def_2 must be limited before the meaning of the elided sentence is derived. Because the restriction of (def2 x) contains no free variables, it is replaced by a function with an empty argument list, as shown in 167.

Example 167

```
Fred chased the cat.

((def1) | (name (def1) Fred)), λ(x)(and (chase x ((def2 x) | (cat (def2 x))))
                                        (= (def2 x) (def3)))
```

Because of the meaning of equality and the meaning of the vertical bar, the formula in 167 is equivalent to the following:

Example 168

```
Fred chased the cat.

((def1) | (name (def1) Fred)), λ(x)(and (chase x (def3)) (cat (def3)))
```

Now that the meaning of the trigger sentence is constrained, we can derive the meaning of the elided sentence. Using the representation of the verb phrase in 167, the elided sentence receives the meaning shown in 169.

Example 169

```
George did too.

((def4) | (name (def4) George)),
   λ(x)(and (chase x ((def2 x) | (cat (def2 x))))
            (= (def2 x) (def3)))
; George saw the same cat as Fred did.
```

Notice that this formula can also be reduced, as shown in 170.

Example 170

```
George did too.

((def4) | (name (def4) George)), λ(x)(and (chase x (def3)) (cat (def3)))
; George saw the same cat as Fred did.
```

Because the final meaning for *the cat* is (def_3) in 167 and 169, it must denote the same cat in both the elided and trigger sentence meanings. Hence, we have demonstrated, with a very simple example, how the argument list for a definite function contained in the representation of the verb phrase for a trigger sentence must be limited using the argument reduction constraint before the meaning of the elided verb phrase is derived.

It is quite easy to see that (def_3), in the previous example, denotes the same individual in the trigger and elided sentences since it has no arguments. However, functions with arguments can also denote the same individuals in trigger and elided sentences. Consider example 171.

Example 171

```
Fred told every girl_i about her_i mother.
George did too.
Meanings:
George told the same girls about their mothers.
```

Consider the initial representation of the trigger sentence, shown in 172.

Example 172

```
Fred told every girl about her mother.

((def1) | (name (def1) Fred)),
   λ(x)(tell x [∀y : (girl y) y]
            (about ((def2 x y) | (and (mother (def2 x y))
                                      (possess (her3 x y)
                                               (def2 x y))))))
```

Now given that the antecedent for *her* is *every girl*, the logical form is updated as shown in example

173.

Example 173

```
Fred told every girl_i about her_i mother.

((def1) | (name (def1) Fred)),
   λ(x)(tell x [∀y : (girl y) y]
             (about ((def2 x y) | (and (mother (def2 x y))
                                       (possess (her3 x y)
                                                (def2 x y))
                                       (= (her3 x y) y)))))
```

Now we must apply the argument reduction constraint to $(def_2\ x\ y)$. Notice y is the only variable not bound inside the function's restriction. Hence, $(def_2\ x\ y)$ should be replaced with a function of y, as shown in 174.

Example 174

```
Fred told every girl_i about her_i mother.

((def1) | (name (def1) Fred)),
   λ(x)(and (tell x [∀y : (girl y) y]
                  (about ((def2 x y) | (and (mother (def2 x y))
                                            (possess (her3 x y)
                                                     (def2 x y))
                                            (= (her3 x y) y)))))
            (= (def2 x y) (def4 y)))
```

Now, the meaning of the elided sentence can be derived using the verb phrase in 174.

Example 175

```
George did too.

((def1) | (name (def1) Fred)),
   λ(x)(and (tell x [∀y : (girl y) y]
                  (about ((def2 x y) | (and (mother (def2 x y))
                                            (possess (her3 x y)
                                                     (def2 x y))
                                            (= (her3 x y) y)))))
            (= (def2 x y) (def4 y)))
; George told the same girls about their mothers.
```

The function representing *her mother*, $(def_4\ y)$, denotes the same group of individuals in the elided and trigger sentences, despite the fact that y is contained in the argument list. A definite function denotes the same individual in the trigger and elided sentences if its argument list does not contain the variable corresponding to the lambda operator abstracting the subject from the trigger verb phrase, any existentially quantified variables, or any universal variables whose restriction contains a lambda or existentially quantified variable. When the argument list of a definite function in a trigger verb phrase contains an existentially quantified variable, that function may denote different individuals in the elided and trigger sentences[14]. When the argument list of a definite function

[14]The definite noun phrase could also denote the same individual in the elided and trigger sentences. This ambiguity

in the trigger verb phrase contains the lambda variable corresponding to the subject of the verb phrase, then following lambda substitution, the argument lists of the functions differ. Finally, when the argument list of a definite function contains a universal variable whose restriction contains a lambda or existentially quantified variable, then the domain of the universal operator can be different in the trigger and elided sentences. Hence, the definite function may denote different individuals in the trigger and elided sentences.

In the rest of this section, we discuss two more examples of verb phrase ellipsis. These examples were already discussed in Chapter two, and are discussed here to demonstrate that we can also handle them with our general definite representation. First, consider how we handle example 6.

Example 6

Trigger Sentence: Fred$_i$ loves his$_i$ wife.
Elided Sentence: George$_j$ does too.
Meanings:
1. George loves Fred's wife.
2. George loves George's wife.

The initial representation of the trigger sentence is shown in 176.

Example 176

Fred loves his wife.

$((\text{def}_1) \mid (\text{name } (\text{def}_1) \text{ Fred}))$,
$\lambda(x)(\text{love } x \; ((\text{def}_2 \; x) \mid (\text{and } (\text{wife } (\text{def}_2 \; x)) (\text{possess } (\text{his}_3 \; x) \; (\text{def}_2 \; x)))))$

Suppose the pronoun resolution module decides that the antecedent for *his* is *Fred*, then the logical form for the trigger sentence is modified, as indicated in 177.

Example 177

Fred$_i$ loves his$_i$ wife.

$((\text{def}_1) \mid (\text{name } (\text{def}_1) \text{ Fred}))$,
$\lambda(x)(\text{love } x \; ((\text{def}_2 \; x) \mid (\text{and } (\text{wife } (\text{def}_2 \; x)) (\text{possess } (\text{his}_3 \; x) \; (\text{def}_2 \; x)) (\text{or } (= (\text{his}_3 \; x) \; (\text{def}_1)) (= (\text{his}_3 \; x) \; x)))))$

Given that we can pick the intended meaning for the pronoun *his*, we can derive two different readings of the elided sentence.

Assume that *his* refers indirectly to *Fred*. In this case, the intended meaning for the trigger sentence is shown in 178.

arises because indefinites are ambiguous. They can either denote the same individual in the trigger and elided sentences, or they can denote different individuals. We discuss this ambiguity in Chapter four.

Example 178

Fred$_i$ loves his$_i$ wife.

```
((def1) | (name (def1) Fred)),
   λ(x)(love x ((def2 x) | (and (wife (def2 x))
                                (possess (his3 x) (def2 x))
                                (= (his3 x) x))))
```

Notice that def_2 is a function of the lambda variable x. Also notice that the pronoun function contained in the restriction of the definite function (i.e., (his_3 x)) is equated with the lambda variable x. Because the variable x is unbound in the restriction for (def_2 x), that function is not changed by the argument reduction constraint. Hence, the representation of the verb phrase in 178 is used to derive the sloppy reading of the elided sentence, as shown in 179.

Example 179

George does too.

```
((def5) | (name (def5) George)),
   λ(x)(love x ((def2 x) | (and (wife (def2 x))
                                (possess (his3 x) (def2 x))
                                (= (his3 x) x))))
; George loves George's wife
```

Notice that the function def_2 denotes different individuals in the trigger and elided sentences, depending on the value substituted for x. Hence, we are able to derive the sloppy reading of the elided sentence of example 6.

On the other hand, suppose that *his* refers directly to *Fred*. Then the intended meaning of the trigger sentence is shown in 180.

Example 180

Fred$_i$ loves his$_i$ wife.

```
((def1) | (name (def1) Fred)),
   λ(x)(love x ((def2 x) | (and (wife (def2 x))
                                (possess (his3 x) (def2 x))
                                (= (his3 x) (def1)))))
```

Notice that once the pronoun function is replaced by (def_1), the restriction of (def_2 x) contains no free variables except those in the argument list of the function itself. Because of this, we replace (def_2 x) with a function whose argument list is empty, as shown in 181.

Example 181

```
Fred_i loves his_i wife.

((def1) | (name (def1) Fred)),
   λ(x)(and (love x ((def2 x) | (and (wife (def2 x))
                                     (possess (his3 x) (def2 x))
                                     (= (his3 x) (def1)))))
            (= (def2 x) (def4)))
```

Using the representation of the verb phrase in 181, we are able to derive the strict reading of the elided sentence, as shown in 182.

Example 182

```
George does too.

((def5) | (name (def5) George)),
   λ(x)(and (love x ((def2 x) | (and (wife (def2 x))
                                     (possess (his3 x) (def2 x))
                                     (= (his3 x) (def1)))))
            (= (def2 x) (def4)))
; George loves Fred's wife.
```

Notice that (def_4) is a constant, and so denotes the same individual in the meanings of the trigger and elided sentences. Hence, our general representation of definite noun phrases in logical form allows us to derive both the sloppy and strict readings of the elided sentence in example 6.

Next, consider example 80.

Example 80

```
Trigger Sentence: Fred_i showed (his_i mother)_j her_j dog.
Elided Sentence:  George_i did too.
Meanings:
1. George showed Fred's mother Fred's mother's dog.
2. George showed George's mother George's mother's dog.
3. *George showed George's mother Fred's mother's dog.
4. *George showed Fred's mother George's mother's dog.
```

The trigger sentence is initially represented as shown in 183.

Example 183

```
Fred showed his mother her dog.

((def1) | (name (def1) Fred)),
   λ(x)(show x ((def4 x) | (and (dog (def4 x))
                                (possess (her5 x) (def4 x))))
               ((def2 x) | (and (mother (def2 x))
                                (possess (his3 x) (def2 x)))))
```

Given that the antecedent for *his* is *Fred* and the antecedent for *her* is *his mother*, we update the logical form, as shown in 184.

Example 184

```
Fred_i showed (his_i mother)_j her_j dog.

((def1) | (name (def1) Fred)),
   λ(x)(show x ((def4 x) | (and (dog (def4 x))
                                (possess (her5 x) (def4 x))
                                (= (her5 x) (def2 x))))
               ((def2 x) | (and (mother (def2 x))
                                (possess (his3 x) (def2 x))
                                (or (= (his3 x) x)
                                    (= (his3 x) (def1))))))
```

Before deriving the meaning of the elided sentence, we need to determine the intended meaning of (his_3 x). Depending on the meaning we choose, we provide the two possible readings of the elided sentence.

Suppose the pronoun *his* refers indirectly to the syntactic subject of the sentence. Then, the intended meaning of the trigger sentence is shown in 185.

Example 185

```
Fred_i showed (his_i mother)_j her_j dog.

((def1) | (name (def1) Fred)),
   λ(x)(show x ((def4 x) | (and (dog (def4 x))
                                (possess (her5 x) (def4 x))
                                (= (her5 x) (def2 x))))
               ((def2 x) | (and (mother (def2 x))
                                (possess (his3 x) (def2 x))
                                (= (his3 x) x))))
```

Because the restriction of (def_2 x) contains the variable x, that function is not changed by the argument reduction constraint. Similarly, because the restriction of (def_4 x) contains the function (def_2 x) (which must be a function of x), that function is unchanged by the argument reduction constraint. Given the representation of the verb phrase in 185, we derive the sloppy meaning for the elided sentence, as shown in 186.

Example 186

```
George did too.

((def8) | (name (def8) George)),
   λ(x)(show x ((def4 x) | (and (dog (def4 x))
                                (possess (her5 x) (def4 x))
                                (= (her5 x) (def2 x))))
               ((def2 x) | (and (mother (def2 x))
                                (possess (his3 x) (def2 x))
                                (= (his3 x) x))))
; George showed George's mother George's mother's dog
```

Notice that both def_2 and def_4 denote different individuals in the trigger and elided sentences,

depending on the value substituted for x.

On the other hand, suppose that *his* refers directly to the noun phrase *Fred*. Then the intended meaning of the trigger sentence follows:

Example 187

```
Fred_i showed (his_i mother)_j her_j dog.

((def1) | (name (def1) Fred)),
   λ(x)(show x ((def4 x) | (and (dog (def4 x))
                                (possess (her5 x) (def4 x))
                                (= (her5 x) (def2 x))))
               ((def2 x) | (and (mother (def2 x))
                                (possess (his3 x) (def2 x))
                                (= (his3 x) (def1)))))
```

Consider the restrictions on (def_2 x) and (def_4 x). In the restriction of (def_2 x), the pronoun function (his_3 x) is equated with (def_1). Because equality is like replacement, the restriction of (def_2 x) contains no free variables. Hence, the argument reduction constraint replaces that function with another function whose argument list is empty (say (def_6)), as shown in 188.

Example 188

```
Fred_i showed (his_i mother)_j her_j dog_k.

((def1) | (name (def1) Fred)),
   λ(x)(and (show x ((def4 x) | (and (dog (def4 x))
                                     (possess (her5 x) (def4 x))
                                     (= (her5 x) (def2 x))))
                    ((def2 x) | (and (mother (def2 x))
                                     (possess (his3 x) (def2 x))
                                     (= (his3 x) (def1)))))
            (= (def2 x) (def6)))
```

Because (her_5 x) is replaced by (def_2 x) and (def_2 x) is replaced by (def_6), the restriction of (def_4 x) contains no variables. Hence, it is replaced with a function whose argument list is empty (say (def_7)).

Example 189

```
Fred_i showed (his_i mother)_j her_j dog_k.

((def1) | (name (def1) Fred)),
   λ(x)(and (show x ((def4 x) | (and (dog (def4 x))
                                     (possess (her5 x) (def4 x))
                                     (= (her5 x) (def2 x))))
                    ((def2 x) | (and (mother (def2 x))
                                     (possess (his3 x) (def2 x))
                                     (= (his3 x) (def1)))))
            (= (def2 x) (def6))
            (= (def4 x) (def7)))
```

Given the meaning of equality and the meaning of the restrictions of functions, the formula in

189 can be simplified as shown in 190.

Example 190

```
Fred_i showed (his_i mother)_j her_j dog_k.

((def1) | (name (def1) Fred)), λ(x)(and (show x (def7) (def6))
                                        (mother (def6))
                                        (possess (def1) (def6))
                                        (dog (def7))
                                        (possess (def6) (def7)))
```

We derive the strict meaning of the elided sentence (shown in 191) using the verb phrase from the representation of the trigger sentence in 189.

Example 191

```
George did too.

((def8) | (name (def8) George)),
   λ(x)(and (show x ((def4 x) | (and (dog (def4 x))
                                     (possess (her5 x) (def4 x))
                                     (= (her5 x) (def2 x))))
                    ((def2 x) | (and (mother (def2 x))
                                     (possess (his3 x) (def2 x))
                                     (= (his3 x) (def1)))))
            (= (def2 x) (def6))
            (= (def4 x) (def7)))
; George showed Fred's mother Fred's mother's dog.
```

The meaning of the elided sentences can also be simplified as shown in 192.

Example 192

```
George did too.

((def8) | (name (def8) George)), λ(x)(and (show x (def7) (def6))
                                          (mother (def6))
                                          (possess (def1) (def6))
                                          (dog (def7))
                                          (possess (def6) (def7)))
; George showed Fred's mother Fred's mother's dog
```

Compare the meaning of the trigger sentence in 190 with the meaning of the elided sentence in 192. Notice, that (def_6) denotes the same individual in both the trigger and elided sentence representations, as expected. Additionally, (def_7) denotes the same individual in both representations. Thus, the strict meaning is provided using our approach.

If a definite function is contained in a trigger verb phrase, we must limit the meaning of that definite function using pronoun resolution or the argument reduction constraint (depending on whether the definite is anaphoric or not) before the meaning of the elided sentence is provided. Once the meaning of the definite is determined, then we can derive the meaning of the elided sentence. Using this approach, we are able to derive both strict and sloppy readings for elided

sentences.

3.3.5 Definite Noun Phrase Examples from Chapter Two

Now that we have provided a general representation for definite noun phrases, a method to update logical form once anaphora resolution has occurred, and a strategy to handle verb phrase ellipsis, we turn our attention to the examples of Chapter two that provided difficulties for previous approaches (i.e., examples 94, 95, 38 and 106).

Consider example 94. As discussed in Chapter two, this example presents problems for Reinhart's [40] approach. Because *his mother* does not c-command *her* (using Reinhart's definition of c-command discussed in Section 2.2.3), the definite cannot bind the pronoun. However, given that the antecedent for *his* is *every man*, *his mother* is not referential. Hence, Reinhart's model cannot handle this example.

Example 94

Every man$_i$ discussed the behavior of (his$_i$ mother)$_j$ with her$_j$ psychiatrist.

However, our approach handles example 94, as we will now show. Consider the initial representation of this sentence, shown in 193.

Example 193

Every man discussed the behavior of his mother with her psychiatrist.

```
∀x: (man x)
 x, λ(y)(discuss y ((def1 y) | (and (behavior (def1 y))
                                    (of (def1 y)
                                        ((def2 y) |
                                           (and (mother (def2 y))
                                                (possess (his3 y)
                                                         (def2 y)))))))
                 (with ((def4 y) | (and (psychiatrist (def4 y))
                                        (possess (her5 y) (def4 y))))))
```

Given that the antecedent for *his* is *every man* and the antecedent for *her* is *his mother*, we can update our logical form, as shown in 194.

Example 194

Every man$_i$ discussed the behavior of (his$_i$ mother)$_j$ with her$_j$ psychiatrist.

```
∀x: (man x)
 x, λ(y)(discuss y ((def1 y) | (and (behavior (def1 y))
                                    (of (def1 y)
                                        ((def2 y) |
                                           (and (mother (def2 y))
                                                (possess (his3 y)
                                                         (def2y))
                                                (or (= (his3 y) y)
                                                    (= (his3 y) x)))))))
                 (with ((def4 y) | (and (psychiatrist (def4 y))
                                        (possess (her5 y) (def4 y))
                                        (= (her5 y) (def2 y))))))
```

If we pick the intended meaning of (his$_3$ y) (either meaning is fine) and apply the argument reduction constraint to the definite functions, our approach provides a reasonable meaning for the sentence in example 94.

Now consider example 95.

Example 95

```
Fredi discussed the behavior of (hisi mother)j with herj psychiatrist.
Billk did too.
Meanings:
1. Bill discussed the behavior of Bill's mother with Bill's mother's
   psychiatrist.
2. Bill discussed the behavior of Fred's mother with Fred's mother's
   psychiatrist.
3. *Bill discussed the behavior of Bill's mother with Fred's mother's
   psychiatrist.
4. *Bill discussed the behavior of Fred's mother with Bill's mother's
   psychiatrist.
```

Reinhart's model provides meanings 1 and 3 of the elided sentence, instead of meanings 1 and 2. However, our approach provides the two expected readings of the elided sentence, as we will show. The initial representation of the trigger sentence is shown in 195.

Example 195

```
Fred discussed the behavior of his mother with her psychiatrist.

((def1) | (name (def1) Fred)),
   λ(y)(discuss y ((def2 y) |
                     (and (behavior (def2 y))
                          (of (def2 y)
                              ((def3 y) |
                                 (and (mother (def3 y))
                                      (possess (his4 y) (def3 y)))))))
                (with ((def5 y) | (and (psychiatrist (def5 y))
                                       (possess (her6 y) (def5 y))))))
```

We must determine the antecedents for the pronouns in the trigger sentence before the verb phrase is used to provide the meaning of the elided sentence. Given that the antecedent for ***his*** is ***Fred*** and the antecedent for ***her*** is ***his mother***, the logical form is updated as shown in 196.

Example 196

Fred$_i$ discussed the behavior of (his$_i$ mother)$_j$ with her$_j$ psychiatrist.

```
((def1) | (name (def1) Fred)),
 λ(y)(discuss y ((def2 y) | (and (behavior (def2 y))
                                 (of (def2 y)
                                     ((def3 y) |
                                        (and (mother (def3 y))
                                             (possess (his4 y) (def3 y))
                                             (or (= (his4 y) y)
                                                 (= (his4 y) (def1))))))))
              (with ((def5 y) | (and (psychiatrist (def5 y))
                                     (possess (her6 y) (def5 y))
                                     (= (her6 y) (def3 y))))))
```

The representation of the trigger sentence in 196 is ambiguous. We must determine the intended meaning of the pronoun function (his$_4$ y) before we derive the meaning of the elided sentence.

Suppose that we decide that ***his*** refers indirectly to *Fred*, then the preferred meaning of the trigger sentence is shown in 197.

Example 197

Fred$_i$ discussed the behavior of (his$_i$ mother)$_j$ with her$_j$ psychiatrist.

```
((def1) | (name (def1) Fred)),
   λ(y)(discuss y ((def2 y) | (and (behavior (def2 y))
                                   (of (def2 y)
                                       ((def3 y) |
                                          (and (mother (def3 y))
                                               (possess (his4 y) (def3 y))
                                               (= (his4 y) y))))))
                (with ((def5 y) | (and (psychiatrist (def5 y))
                                       (possess (her6 y) (def5 y))
                                       (= (her6 y) (def3 y))))))
```

Next, we must apply the argument reduction constraint to each definite function in 197. Since the restriction of (def$_3$ y) contains the variable *y*, it must remain a function of *y*. Additionally, because the restriction of (def$_2$ y) contains (def$_3$ y), (def$_2$ y) is unchanged by argument reduction. Finally, because the restriction of (def$_5$ y) contains the pronoun function (her$_6$ y) (which is replaced by (def$_3$ y)), (def$_5$ y) is unchanged. Because the logical form in 197 is not affected by the argument reduction constraint, we derive the sloppy meaning for the elided sentence (shown in 198) using the representation of the verb phrase in 197.

Example 198

George did too.

```
((def7) | (name (def7) George)),
   λ(y)(discuss y ((def2 y) | (and (behavior (def2 y))
                                   (of (def2 y)
                                       ((def3 y) |
                                          (and (mother (def3 y))
                                               (possess (his4 y) (def3 y))
                                               (= (his4 y) y))))))
                (with ((def5 y) | (and (psychiatrist (def5 y))
                                       (possess (her6 y) (def5 y))
                                       (= (her6 y) (def3 y))))))
; George discussed the behavior of George's mother with George's mother's
; psychiatrist
```

Hence, we are able to derive the sloppy reading for the elided sentence in example 95.

Now suppose that *his* refers directly to *Fred*, then the preferred meaning of the trigger sentence is shown in 199.

Example 199

Fred$_i$ discussed the behavior of (his$_i$ mother)$_j$ with her$_j$ psychiatrist.

```
((def1) | (name (def1) Fred)),
   λ(y)(discuss y ((def2 y) | (and (behavior (def2 y))
                                   (of (def2 y)
                                       ((def3 y) |
                                          (and (mother (def3 y))
                                               (possess (his4 y) (def3 y))
                                               (= (his4 y) (def1)))))))
                (with ((def5 y) | (and (psychiatrist (def5 y))
                                       (possess (her6 y) (def5 y))
                                       (= (her6 y) (def3 y))))))
```

Next, we apply the argument reduction constraint to each definite function in 199. Because the restriction of (def$_3$ y) contains no unbound variables (except in its argument list), we replace that function with a function with no arguments (say (def$_8$)). We also replace (def$_2$ y) with a function whose argument list is empty (since (def$_3$ y) is replaced with (def$_8$)). Finally, we replace (def$_5$ y) with a function with no arguments. This reading is shown in 200.

Example 200

Fred$_i$ discussed the behavior of (his$_i$ mother)$_j$ with her$_j$ psychiatrist.

```
((def1) | (name (def1) Fred)),
   λ(y)(and (discuss y ((def2 y) | (and (behavior (def2 y))
                                        (of (def2 y)
                                            ((def3 y) |
                                               (and (mother (def3 y))
                                                    (possess (his4 y)
                                                             (def3 y))
                                                    (= (his4 y) (def1)))))
                                        (= (def3 y) (def8))))
                    (with ((def5 y) | (and (psychiatrist (def5 y))
                                           (possess (her6 y) (def5 y))
                                           (= (her6 y) (def3 y))))))
            (= (def2 y) (def9))
            (= (def5 y) (def10)))
```

The strict meaning of the elided sentence (shown in 201) is derived using the representation of the verb phrase from 200.

Example 201

George did too.

```
((def7) | (name (def7) George)),
   λ(y)(and (discuss y ((def2 y) | (and (behavior (def2 y))
                                        (of (def2 y)
                                            ((def3 y) |
                                               (and (mother (def3 y))
                                                    (possess (his4 y)
                                                             (def3 y))
                                                    (= (his4 y) (def1)))))
                                        (= (def3 y) (def8))))
                    (with ((def5 y) | (and (psychiatrist (def5 y))
                                           (possess (her6 y) (def5 y))
                                           (= (her6 y) (def3 y))))))
            (= (def2 y) (def9))
            (= (def5 y) (def10)))
; George discussed the behavior of Fred's mother with Fred's mother's
; psychiatrist
```

Hence, our approach provides both of the possible meanings of the elided sentence in example 95.

Next, we discuss how our approach handles two examples that presented problems for Partee and Bach's [38] model. First, consider example 38.

Example 38

Every man$_i$ gave the psychiatrist who cares for (his$_i$ mother)$_j$ her$_j$ diary.

Our approach handles this example, but not in as straightforward a manner as the previous examples. In all of the examples discussed so far, anaphoric information is added to the logical form directly without applying the argument reduction constraint to definite functions. To handle example 38, we must modify our initial approach. Consider our initial representation for the sentence in 38, shown in 202.

Example 202

```
Every man gave the psychiatrist who cares for his mother her diary.

∀x: (man x)
 x, λ(y)(give y ((def4 y) | (and (diary (def4 y))
                                 (possess (her5 y) (def4 y))))
              ((def1 y) |
                 (and (psychiatrist (def1 y))
                      (def1 y),
                        λ(z)(give z ((def2 y z) |
                                       (and (mother (def2 y z))
                                            (possess (his3 y z)
                                                     (def2 y z))))))))
```

Notice the initial representation of *his mother* (shown in 202) contains a variable not included in the argument list of the pronoun function for *her*. Suppose that the pronoun resolution module decides that the antecedent for *his* is *every man* and the antecedent for *her* is *his mother*. We can certainly add the information about the antecedent for *his* to the logical form in 202. However, we cannot immediately add the information indicating the antecedent for *her*. To add this information, we would have to equate (her_5 y) with (def_2 y z). Given that a pronoun function must be updated within its lambda environment, this update would not provide a well-formed meaning. The variable z would be unbound. However, all is not lost. Let us update the logical form to reflect the fact that *every man* is the antecedent for *his*, remembering that *his mother* is the antecedent for *her*.

Example 203

```
Every man_i gave the psychiatrist who cares for his_i mother her diary.

∀x: (man x)
 x, λ(y)(give y ((def4 y) | (and (diary (def4 y))
                                 (possess (her5 y) (def4 y))))
              ((def1 y) |
                 (and (psychiatrist (def1 y))
                      (def1 y),
                        λ(z)(give z ((def2 y z) |
                                       (and (mother (def2 y z))
                                            (possess (his3 y z)
                                                     (def2 y z))
                                            (or (= (his3 y z) y)
                                                (= (his3 y z) x))))))))
```

Notice that the pronoun function associated with *her* (i.e., (her_5 y)) is a function of y. We can legally equate (her_5 y) with a function of y, a function of z, or a function with no arguments. Examine the restriction on the function representing *his mother*, that is (def_2 y z). Depending on the value we pick for the pronoun function (his_3 y z), we replace (def_2 y z) with either a function of y or a

function of x (either of which is accessible to (her_5 y)). Assume that we pick the first meaning of the pronoun (i.e., (= (his_3 y z) y)). Given the argument reduction constraint, (def_2 y z) is replaced by a function of y, say (def_6 y). Since (def_6 y) is compatible with the pronoun function (her_5 y), we can now add the information that the antecedent for ***her*** is ***his mother***. These updates are shown in 204.

Example 204

Every man_i gave the psychiatrist who cares for (his_i mother)$_j$ her_j diary.

```
∀x: (man x)
 x, λ(y)(give y ((def4 y) | (and (diary (def4 y))
                                 (possess (her5 y) (def4 y))
                                 (= (her5 y) (def6 y)))))
              ((def1 y) |
                 (and (psychiatrist (def1 y))
                      (def1 y),
                        λ(z)(and (give z ((def2 y z) |
                                          (and (mother (def2 y z))
                                               (possess (his3 y z)
                                                        (def2 y z))
                                               (= (his3 y z) y)))
                                 (= (def2 y z) (def6 y)))))
```

Hence, we are able to provide the meaning of the sentence in 38.

To some, this relaxation step may seem like unnecessary work, perhaps signaling a bad representation for definites. However, when a definite is embedded in a relative clause, it may be subject to some influences that make it incompatible with a pronoun in the matrix sentence. Not all definite noun phrases embedded in a relative clause should be accessible to a pronoun in the matrix sentence. Consider, for example:

Example 205

*Every $woman_i$ gave the psychiatrist who told every man_j about (his_j mother)$_k$ her_k diary.

Despite the fact that *his mother* is a definite, it should not be the antecedent for *her* when *every man* is the antecedent for *his*. In fact, using our approach the anaphoric link between *his mother* and *her* (given the co-indexing in the sentence) is not allowed. Consider our initial representation for this sentence (shown in 206).

Example 206

Every woman gave the psychiatrist who told every man about his mother her diary.

```
∀x: (woman x)
 x, λ(y)(give y ((def1 y) | (and (diary (def1 y))
                                 (possess (her2 y) (def1 y))))
                ((def3 y) |
                   (and (psychiatrist (def3 y))
                        (def3 y),
                          λ(z)(tell z [∀w: (man w) w]
                                    (about ((def4 y z w) |
                                              (and (mother (def4 y z w))
                                                   (possess
                                                     (his5 y z w)
                                                     (def4 y z w)))))))))
```

We cannot immediately assert that *his mother* is the antecedent for *her*, but we can add the information that the antecedent for *his* is *every man*, as shown in 207.

Example 207

Every woman gave the psychiatrist who told every man$_j$ about his$_j$ mother her diary.

```
∀x: (woman x)
 x, λ(y)(give y ((def1 y) | (and (diary (def1 y))
                                 (possess (her2 y) (def1 y))))
                ((def3 y) |
                   (and (psychiatrist (def3 y))
                        (def3 y),
                          λ(z)(tell z [∀w: (man w) w]
                                    (about ((def4 y z w) |
                                              (and (mother (def4 y z w))
                                                   (possess (his5 y z w)
                                                            (def4 y z w))
                                                   (= (his5 y z w) w)))))))))
```

When we can apply the argument reduction constraint to (def$_4$ y z w), it is replaced by a function of w (say (def$_6$ w)). However, since *her* is represented as a function of y, we cannot equate that pronoun function (in its lambda environment) with the reduced function for *his mother* (i.e., (def$_6$ w)). Hence, we are unable to represent the meaning of the sentence in 205, given the co-indexing indicated on the noun phrases. This is precisely the result we want. However, if the antecedent for *his* was some noun phrase outside of the sentence, then the result would have been different. Hence, a definite noun phrase embedded in a relative clause can be the antecedent for a pronoun if the definite function's argument list is compatible with the pronoun function's argument list after argument reduction.

Our approach also handles example 106.

Example 106

Fred$_i$ gave the psychiatrist who cares for (his$_i$ mother)$_j$ her$_j$ diary.
George$_k$ did too.
Meanings:
1. **George gave the psychiatrist who cares for Fred's mother Fred's mother's diary.**
2. **George gave the psychiatrist who cares for George's mother George's mother's diary.**
3. ***George gave the psychiatrist who cares for George's mother Fred's mother's diary.**
4. ***George gave the psychiatrist who cares for Fred's mother George's mother's diary.**

As shown in Chapter two, this example presents a problem for Partee and Bach's approach to verb phrase ellipsis. In particular, they cannot provide the two expected readings of the elided sentence. Their model provides readings 1 and 3 in example 106). However, our approach provides the two expected meanings of the elided sentence. The initial representation of the trigger sentence is shown in 208.

Example 208

```
Fred gave the psychiatrist who cares for his mother her diary.

((def1) | (name (def1) Fred),
   λ(y)(give y ((def5 y) | (and (diary (def5 y))
                                (possess (her6 y) (def5 y))))
            ((def2 y) |
               (and (psychiatrist (def2 y))
                    (def2 y),
                      λ(z)(give z ((def3 y z) |
                                   (and (mother (def3 y z))
                                        (possess (his4 y z)
                                                 (def3 y z))))))))
```

Suppose that the pronoun resolution module informs us that the antecedent for *his* is *Fred* and the antecedent for that *her* is *his mother*. We can directly add the information about the pronoun *his* to the logical form. However, we cannot immediately add the information about the pronoun *her* (otherwise we would have to equate (her$_6$ y) with (def$_3$ y z), resulting in an ill-formed logical form). However, let us update the logical form to indicate that the antecedent for *his* is *Fred*.

Example 209

Fred$_i$ gave the psychiatrist who cares for his$_i$ mother her diary.

```
((def1) | (name (def1) Fred),
   λ(y)(give y ((def5 y) | (and (diary (def5 y))
                                (possess (her6 y) (def5 y))))
            ((def2 y) |
               (and (psychiatrist (def2 y))
                    (def2 y),
                      λ(z)(give z ((def3 y z) |
                                     (and (mother (def3 y z))
                                          (possess (his4 y z)
                                                   (def3 y z))
                                          (or (= (his4 y z) y)
                                              (= (his4 y z)
                                                 (def1))))))))
```

Notice that the argument list of (def$_3$ y z) can be limited to be a function of y or a function with no arguments, depending on the value chosen for the pronoun function (his$_4$ y z).

Suppose that *his* refers to *Fred* indirectly, then the pronoun function (his$_4$ y z) is equated with y, as shown in 210.

Example 210

Fred$_i$ gave the psychiatrist who cares for (his$_i$ mother)$_j$ her$_j$ diary.

```
((def1) | (name (def1) Fred),
   λ(y)(give y ((def5 y) | (and (diary (def5 y))
                                (possess  (her6 y) (def5 y))))
            ((def2 y) |
               (and (psychiatrist (def2 y))
                    (def2 y),
                      λ(z)(and (give z ((def3 y z) |
                                          (and (mother (def3 y z))
                                               (possess (his4 y z)
                                                        (def3 y z))
                                               (= (his4 y z) y))))))))
```

Given this choice, we apply the argument reduction constraint to (def$_3$ y z), limiting it to a function of y. Given this information, the meaning of the trigger sentence is shown in 211.

Example 211

Fred$_i$ gave the psychiatrist who cares for (his$_i$ mother)$_j$ her$_j$ diary.

```
((def1) | (name (def1) Fred),
   λ(y)(give y ((def5 y) | (and (diary (def5 y))
                                (possess  (her6 y) (def5 y))
                                (= (her6 y) (def7 y))))
            ((def2 y) |
               (and (psychiatrist (def2 y))
                    (def2 y),
                      λ(z)(and (give z ((def3 y z) |
                                        (and (mother (def3 y z))
                                             (possess (his4 y z)
                                                      (def3 y z))
                                             (= (his4 y z) y))))
                               (= (def3 y z) (def7 y))))))
```

Now we must apply the argument reduction constraint to (def_2 y) and (def_5 y). Since the restriction of each function contains the function (def_7 y), they remain functions of *y*. Hence, the representation of the verb phrase in 211 is used to provide the sloppy reading of the elided sentence (shown in 212).

Example 212

```
George did too.

((def8) | (name (def8) George),
   λ(y)(give y ((def5 y) | (and (diary (def5 y))
                                (possess  (her6 y) (def5 y))
                                (= (her6 y) (def7 y))))
            ((def2 y) |
               (and (psychiatrist (def2 y))
                    (def2 y),
                      λ(z)(and (give z ((def3 y z) |
                                        (and (mother (def3 y z))
                                             (possess (his4 y z)
                                                      (def3 y z))
                                             (= (his4 y z) y))))
                               (= (def3 y z) (def7 y))))))
; George gave the psychiatrist who cares for George's mother George's
; mother's diary.
```

On the other hand, suppose that *his* refers directly to *Fred*, then the pronoun function (his_4 y z) is equated with (def_1), as shown in 213.

Example 213

Fred$_i$ gave the psychiatrist who cares for (his$_i$ mother)$_j$ her$_j$ diary.

```
((def1) | (name (def1) Fred),
   λ(y)(give y ((def5 y) | (and (diary (def5 y))
                                (possess  (her6 y) (def5 y))))
            ((def2 y). |
               (and (psychiatrist (def2 y))
                    (def2 y),
                  λ(z)(and (give z ((def3 y z) |
                                    (and (mother (def3 y z))
                                         (possess (his4 y z)
                                                  (def3 y z))
                                         (= (his4 y z) (def1)))))))))
```

Given this choice, we apply the argument reduction constraint to (def$_3$ y z), limiting it to a function with no arguments. Given this update, the meaning of the trigger sentence is shown in 214.

Example 214

Fred$_i$ gave the psychiatrist who cares for (his$_i$ mother)$_j$ her$_j$ diary.

```
((def1) | (name (def1) Fred),
   λ(y)(give y ((def5 y) | (and (diary (def5 y))
                                (possess  (her6 y) (def5 y))
                                (= (her6 y) (def8))))
            ((def2 y) |
               (and (psychiatrist (def2 y))
                    (def2 y),
                  λ(z)(and (give z ((def3 y z) |
                                    (and (mother (def3 y z))
                                         (possess (his4 y z)
                                                  (def3 y z))
                                         (= (his4 y z) (def1)))))))
                                (= (def3 y z) (def8))))))
```

Because (def$_3$ y z) is replaced by a function with no arguments, so can (def$_2$ y). Also, because the pronoun function (her$_6$ y) is replaced by (def$_8$), (def$_5$ y) can be replaced by a function with no arguments. These updates are shown in 215.

Example 215

Fred$_i$ gave the psychiatrist who cares for (his$_i$ mother)$_j$ her$_j$ diary.

```
((def1) | (name (def1) Fred),
   λ(y)(and (give y ((def5 y) | (and (diary (def5 y))
                                     (possess (her6 y) (def5 y))
                                     (= (her6 y) (def8))))

                ((def2 y) |
                   (and (psychiatrist (def2 y))
                        (def2 y),
                          λ(z)(and (give z
                                        ((def3 y z) |
                                           (and (mother (def3 y z))
                                                (possess (his4 y z)
                                                         (def3 y z))
                                                (= (his4 y z)
                                                   (def1)))))
                                   (= (def3 y z) (def8))))))
          (= (def2 y) (def9))
          (= (def5 y) (def10)))
```

Given the representation of the trigger sentence indicated in 215, we derive the strict meaning of the elided sentence, as shown in 216.

Example 216

George did too.

```
((def10) | (name (def10) George),
   λ(y)(and (give y ((def5 y) | (and (diary (def5 y))
                                     (possess (her6 y) (def5 y))
                                     (= (her6 y) (def8)))))

                ((def2 y) |
                   (and (psychiatrist (def2 y))
                        (def2 y),
                          λ(z)(and (give z
                                        ((def3 y z) |
                                           (and (mother (def3 y z))
                                                (possess (his4 y z)
                                                         (def3 y z))
                                                (= (his4 y z)
                                                   (def1)))))
                                   (= (def3 y z) (def8))))))
          (= (def2 y) (def9))
          (= (def5 y) (def10)))

; George gave the psychiatrist who cares for Fred's mother Fred's
; mother's diary.
```

Hence, we are able to provide both readings for the elided sentence in example 106. To do so, we limit the argument list of a definite function contained in a relative clause before equating a

pronoun function with it. We handle this type of example in our implementation discussed in Chapter six by gathering information about potential antecedents of pronouns on a need-to-know basis. If a function is immediately compatible with a pronoun function, we do not need to find out more about that function until later. However, if we wish to find out whether a definite function embedded in a relative clause is compatible with a matrix pronoun function, we gather information about the precise meaning of that function.

Now that we have discussed the examples from Chapter two, we turn our attention to several interesting examples. In the next section, we consider argument reduction strategies necessary to provide reasonable meanings for definite donkey sentences[15], Bach-Peters sentences, and Jacobson sentences.

3.3.6 Other Interesting Examples of Definite Noun Phrase Behavior from the Literature

In this section, we examine how our approach handles some examples discussed in the linguistic literature. In particular, we discuss how our approach handles definite donkey sentences, Bach-Peters sentences (originally examined by Bach [2]), and Jacobson sentences (originally discussed in Jacobson [28]).

Consider an example of a definite donkey sentence shown in 217. Because of the complex noun phrase constraint, the definite cannot bind the pronoun in the matrix sentence. However, *his wife* can be the antecedent for *her*.

Example 217

(Every miner who loves (his$_i$ wife)$_j$)$_i$ cherishes her$_j$.

We handle this example using our functional representation of definites and the argument reduction constraint. In particular, the antecedent for *her* can be *his wife*, if after argument reduction, the argument list of the definite function is compatible with the argument list of the pronoun function. The initial logical form representation for the sentence in 217 is shown in 218.

Example 218

Every miner who loves his wife cherishes her.

$\forall$x: (and (miner x)
x, λ(y)(love y ((def$_1$ y) | (and (wife (def$_1$ y))
(possess (his$_2$ y) (def$_1$ y))))))
x, λ(z)(cherish z (her$_3$ z))

Suppose the pronoun resolution module decides that the antecedent for *her* is *his wife*. Because (def$_1$ y) is not compatible with (her$_3$ z), we cannot immediately assert the anaphoric link between *her* and *his wife*. Before we can apply the argument reduction constraint to (def$_1$ y), we must

[15] We use the term *definite donkey sentences* to distinguish them from traditional donkey sentences (donkey sentences were originally examined by Geach [17]). A typical donkey sentence is *Every miner who owns a donkey$_i$ beats it$_i$*. The antecedent for the pronoun *it* is an indefinite. In contrast, example 217 is a definite donkey sentence. The antecedent is a definite noun phrase.

determine the antecedent for *his*[16]. Suppose that the pronoun resolution module informs us that the antecedent for *his* is *every miner*. The logical form is updated as shown in 220.

Example 220

```
(Every miner who loves his_i wife)_i cherishes her.

∀x: (and (miner x)
         x, λ(y)(love y ((def_1 y) | (and (wife (def_1 y))
                                          (possess (his_2 y) (def_1 y))
                                          (or (= (his_2 y) y)
                                              (= (his_2 y) x))))))
 x, λ(z)(cherish z (her_3 z))
```

Notice that (his_2 y) can either be replaced by the variable y or the variable x. If we pick the first alternative (i.e., y), then (def_1 y) must remain a function of y. However, the only things that (her_3 z) can be equated with are the variables x or z, some function of x or z, or a function with no arguments. Hence, given that the antecedent for *her* is *his mother*, we cannot pick the sloppy meaning of *his*[17]. Instead, we must pick the second meaning of (his_2 y) (i.e., x). This update is shown in 221.

Example 221

```
(Every miner who loves his_i wife)_i cherishes her.

∀x: (and (miner x)
         x, λ(y)(love y ((def_1 y) | (and (wife (def_1 y))
                                          (possess (his_2 y) (def_1 y))
                                          (= (his_2 y) x)))))
 x, λ(z)(cherish z (her_3 z))
```

Because the restriction on (def_1 y) contains only the variable x, we replace that function with a function of the variable x, as shown in 222.

Example 222

```
(Every miner who loves his_i wife)_i cherishes her.

∀x: (and (miner x)
         x, λ(y)(and (love y ((def_1 y) | (and (wife (def_1 y))
                                               (possess (his_2 y) (def_1 y))
                                               (= (his_2 y) x))))
                     (= (def_1 y) (def_4 x))))
 x, λ(z)(cherish z (her_3 z))
```

---

[16] Notice that we cannot tell if *his mother* is compatible with *her* until we know how *his* is resolved. This is important, as can be seen by the following example:

Example 219
*Every miner who told every teacher$_i$ about (his$_i$ wife)$_j$ cherishes her$_j$.

In this case, *his wife* cannot be the antecedent for *her*.

[17] We provide the indirect reference to the subject in case the verb phrase of the relative clause is needed to provide the sloppy reading of an elided sentence in verb phrase ellipsis.

Now that (def_1 y) is replaced by (def_4 x), we add the information that *her* depends on *his wife* by equating (her_3 z) with (def_4 x) (which is legal because x is the variable abstracted by the operator, $\lambda(z)$). This update is shown in 223.

Example 223

(Every miner who loves (his_i wife)$_j$)$_i$ cherishes her_j.

```
∀x: (and (miner x)
         x, λ(y)(and (love y ((def1 y) | (and (wife (def1 y))
                                              (possess (his2 y)(def1 y))
                                              (= (his2 y) x))))
                     (= (def1 y) (def4 x)))))
 x,  λ(z)(and (cherish z (her3 z))
              (= (her3 z) (def4 x)))
```

The logical form shown in 223 can be simplified as shown in 224, given the meaning of equality and the restrictions on functions and universals.

Example 224

(Every miner who loves (his_i wife)$_j$)$_i$ cherishes her_j.

```
∀x (if (and (miner x)  x, λ(y)(and (love y (def4 x))
                                   (wife (def4 x))
                                   (possess x (def4 x))))
       x, λ(z)(cherish z (def4 x)))
```

Thus, our approach provides the intended meaning of the sentence in 217.

Next, consider an example similar to 217, except that the anaphor is a definite noun phrase instead of a pronoun. Our approach also handles this type of example.

Example 225

(Every man who loves (his_i mother)$_j$)$_i$ is often exasperated by the old lady$_j$.

The initial representation for this sentence is shown in 226.

Example 226

Every man who loves his mother is often exasperated by the old lady.

```
∀x: (and (man x)
         x, λ(y)(love y ((def1 y) | (and (mother (def1 y))
                                         (possess (his2 y) (def1 y))))))
 x, λ(z)(often-exasperated-by z ((def3 z) | (old-lady (def3 z))))
```

Suppose the antecedent for *his* is *every man* and the antecedent for *the old lady* is *his mother*. We cannot immediately add the information concerning the anaphoric definite, but we can augment the logical form with the pronoun information, as shown in 227.

Example 227

(Every man who loves his$_i$ mother)$_i$ is often exasperated by the old lady.

```
∀x: (and (man x)
         x, λ(y)(love y ((def1 y) | (and (mother (def1 y))
                                         (possess (his2 y) (def1 y))
                                         (or (= (his2 y) y)
                                             (= (his2 y) x)))))
 x, λ(z)(often-exasperated-by z ((def3 z) | (old-lady (def3 z))))
```

Notice that the pronoun *his* can either be equated with the universal variable corresponding to the subject or with the lambda variable corresponding to the head of the relative clause. However, because the antecedent for *the old lady* is *his mother*, the argument lists of (def_1 y) and (def_3 z) must become compatible. Since z is the variable which abstracts x, the functions would be compatible if (def_1 y) is replaced by a function of x. To do this, we must limit the meaning of the pronoun function (his_2 y) to its second meaning (i.e., x) and then replace (def_1 y) with a function of x, say (def_4 x) (shown in 228).

Example 228

(Every man who loves his$_i$ mother)$_i$ is often exasperated by the old lady.

```
∀x: (and (man x)
         x, λ(y)(love y ((def1 y) | (and (mother (def1 y))
                                         (possess (his2 y) (def1 y))
                                         (= (his2 y) x))))
         (= (def1 y) (def4 x)))
 x, λ(z)(often-exasperated-by z ((def3 z) | (old-lady (def3 z))))
```

Because we replace the pronoun function for *his* with the universal variable x, the function corresponding with *his mother* is replaced with a function of x. Since (def_4 x) is compatible with the definite function for *the old lady*, we add the information concerning the anaphoric dependence between *the old lady* and *his mother*, as shown in 229.

Example 229

(Every man who loves (his$_i$ mother)$_j$)$_i$ is often exasperated by the old lady$_j$.

```
∀x: (and (man x)
         x, λ(y)(love y ((def1 y) | (and (mother (def1 y))
                                         (possess (his2 y) (def1 y))
                                         (= (his2 y) x))))
         (= (def1 y) (def4 x)))
 x, λ(z)(and (often-exasperated-by z ((def3 z) | (old-lady (def3 z))))
             (= (def3 z) (def4 x)))
```

Thus, our approach handles definite donkey sentences (i.e., donkey sentences where the antecedent is a definite noun phrase), regardless of whether the anaphoric expression is a definite or a pronoun. Both cases are similar.

Examples 217 and 225 suggest that sometimes additional information about the meanings of noun

phrases embedded in a potential antecedent must be obtained before we can decide the antecedent is compatible with a certain pronoun or anaphoric definite. Based on these two examples, we suggest that definite functions in subject position (as well as those embedded in the subject) should be limited before definite functions in the verb phrase are limited or before pronoun functions in the verb phrase are updated with information about their antecedents. This strategy, which is compatible with Keenan's function principle [31], would allow us to handle examples 217 and 225. There is, however, another type of strategy (which violates the function principle) which is needed to handle Bach-Peters sentences.

Bach-Peters sentences were originally examined by Bach in [2]. An example of a Bach-Peters or crossing reference sentence is shown in example 230.

Example 230

(The boy who wrote her$_j$)$_i$ kissed (the girl who loved him$_i$)$_j$.

This sentence is reasonable for some English speakers, but not all[18]. Given that it is reasonable for some speakers, our approach should handle it. However, we hope that our approach might also shed some light on why the sentence is unreasonable for some English speakers. Consider the initial representation for this sentence, shown in 231.

Example 231

The boy who wrote her kissed the girl who loved him.

```
((def1) | (and (boy (def1))
               (def1), λ(x)(write x (her2 x)))),
   λ(y)(kiss y ((def3 y) | (and (girl (def3 y))
                                (def3 y), λ(z)(love z (him4 y z)))))
```

Now suppose that we are told that the antecedent for *her* is *the girl who loves him* and that the antecedent for *him* is *the boy who wrote her*. To provide the meaning of this sentence, we must apply the argument reduction constraint on (def$_3$ y) (a function in the verb phrase) before determining the antecedent for (her$_2$ x) (a function in the subject). To apply the argument reduction constraint to (def$_3$ y), we must indicate the antecedent for *him* in our logical form (shown in 232).

Example 232

(The boy who wrote her)$_i$ kissed the girl who loved him$_i$.

```
((def1) | (and (boy (def1))
               (def1), λ(x)(write x (her2 x)))),
   λ(y)(kiss y ((def3 y) | (and (girl (def3 y))
                                (def3 y), λ(z)(love z (him4 y z))
                                (or (= (him4 y z) y)
                                    (= (him4 y z) (def1))))))
```

[18] Jacobson [28] claims that *her* can have *the girl who loved him* as its antecedent in 230, but *she* in *The boy who she wrote kissed the girl who loved him* cannot have *the girl who wrote him* as its antecedent. However, there may be pragmatic issues which affect these decisions. For example, *he* in *The model he considered was good enough for the man who took her measurements* can have *the man who took her measurements* as its antecedent and *her* can have *the model he considered* as its antecedent.

For the definite function representing *the girl who loves him* (i.e., (def_3 y)) to become compatible with the pronoun function (her_2 x), we must pick the second meaning for the pronoun function (him_4 y z) (i.e., (= (him_4 y z) (def_1))). Once this choice is made, the function (def_3 y) is replaced by a function with no arguments. These updates are shown in 233.

Example 233

(The boy who wrote her)$_i$ kissed the girl who loved him$_i$.

```
((def1) | (and (boy (def1))
               (def1), λ(x)(write x (her2 x)))),
   λ(y)(and (kiss y ((def3 y) | (and (girl (def3 y))
                                     (def3 y), λ(z)(love z (him4 y z))
                                     (= (him4 y z) (def1)))))
            (= (def3 y) (def5)))
```

Now we can equate (her_2 x) with (def_5) to give the intended meaning of the sentence, as shown in 234.

Example 234

(The boy who wrote her$_j$)$_i$ kissed (the girl who loved him$_i$)$_j$.

```
((def1) | (and (boy (def1))
               (def1), λ(x)(write x (her2 x))
               (= (her2 x) (def5)))),
   λ(y)(and (kiss y ((def3 y) | (and (girl (def3 y))
                                     (def3 y), λ(z)(love z (him4 y z))
                                     (= (him4 y z) (def1)))))
            (= (def3 y) (def5)))
```

These updates predict that if the verb phrase from 234 is the trigger verb phrase for an elided sentence like *Fred did too*, then only the strict reading of the elided sentence should be available (given that *him* refers to subject and *her* refers to *the girl who loves him*). The antecedent for *her* can be *the girl who loves him*, but only if we pick the strict meaning for *him*[19].

To handle Bach-Peters sentences, we must relax the argument list of a function in the verb phrase before equating a pronoun in the subject with that function. This violates the function principal suggested by Keenan [31]. This violation seems to correlate with the fact that the Bach-Peters sentences are harder to understand. If such a relaxation strategy is not allowed in the listener's language, then the sentence in 230 could never receive the interpretation indicated. In fact, many informants find this sentence awkward at best. However, speakers who have the *object-then-subject* relaxation strategy available to them should be able to understand the sentence in 230.

Finally, consider how our approach handles Jacobson sentences [28]. Consider example 235.

Example 235

Everyone$_i$ told (her$_j$ mother)$_k$ that (his$_i$ wife)$_j$ should get a job.

[19]The strict reading may seem odd to the reader. We believe that this observation results because it is unusual for *her* to be anaphorically dependent on *the girl who loved him*, not because the sloppy reading is available given the indices on the noun phrases.

This sentence, though somewhat difficult to understand, can have the meaning indicated by the co-indexing shown in the example. In contrast, consider example 236.

Example 236

```
*(Her_j mother)_k told everyone_i that (his_i wife)_j should get a job.
```

Because *everyone* cannot have scope over the pronoun contained in the subject and because *everyone* distributes over *his wife*, *his wife* cannot be the antecedent for *her*. Hence, the sentence is marked with a star, given the indicated co-indexing of noun phrases. Our approach provides the reading indicated in example 235, and correctly fails to provide the reading indicated in 236.

First, consider how we handle example 235. The initial logical form representation for the sentence is indicated in 237.

Example 237

```
Everyone told her mother that his wife should get a job.

∀x: (person x)
 x, λ(y)(tell y ((def1 y) | (and (mother (def1 y))
                                 (possess (her2 y) (def1 y))))
              [((def3 y)| (and (wife (def3 y))
                               (possess (his4 y) (def3 y)))),
                  λ(z)(get z [∃w: (job w) w])])
```

Given that *everyone* is the antecedent for *his* and *his wife* is the antecedent for *her*, we update the logical form as shown in 238.

Example 238

```
Everyone_i told (her_j mother)_k that (his_i wife)_j should get a job.

∀x: (person x)
 x, λ(y)(tell y ((def1 y)| (and (mother (def1 y))
                                (possess (her2 y) (def1 y))
                                (= (her2 y) (def3 y))))
              [((def3 y) | (and (wife (def3 y))
                                (possess (his4 y) (def3 y))
                                (or (= (his4 y) x)
                                    (= (his4 y) y)))),
                  λ(z)(get z [∃w: (job w) w])])
```

Given that we decide the universal has scope over the existential quantifier, pick one of the meanings of the pronoun *his*, and apply the argument reduction constraint, our approach provides the expected meaning for the sentence in 235. Despite the fact that no relaxation is necessary to indicate the antecedents for the pronouns, the meaning of the sentence is still rather difficult to understand. Perhaps speakers are affected by a leftness constraint (i.e., pronouns cannot be co-indexed with a quantified noun phrase to its right [10,19]) on anaphora. This would certainly explain why this sentence is difficult to understand, even without cross-dependency. However, despite its difficulty, the sentence in 235 is more reasonable than the sentence in 236.

In contrast to 235, our approach cannot provide the indicated meaning of the sentence in 236. To see why, consider the initial logical form representation for the sentence, shown in 239.

Example 239

```
Her mother told everyone that his wife should get a job.

((def1) | (and (mother (def1))
               (possess (her2) (def1)))),
   λ(y)(tell y [∀z: (person z) z]
            [((def3 y z) | (and (wife (def3 y z))
                                (possess (his4 y z) (def3 y z)))),
               λ(x)(get x [∃w: (job w) w])])
```

Given that the antecedent for *his* is *everyone*, we update the logical form as shown in 240.

Example 240

```
Her mother told everyone_i that his_i wife should get a job.

((def1) | (and (mother (def1))
               (possess (her2) (def1)))),
   λ(y)(tell y [∀z: (person z) z]
            [((def3 y z) | (and (wife (def3 y z))
                                (possess (his4 y z) (def3 y z))
                                (= (his4 y z) z))),
               λ(x)(get x [∃w: (job w) w])])
```

However, we cannot update the logical form to indicate that the antecedent for *her* is *his wife*. The function representing *his wife*, (def_3 y z), must be a function of *z*, given the argument reduction constraint. Hence, we cannot equate the pronoun function for *her*, (her_2), with the definite function (without violating the formal consistency constraint).

Thus, our approach handles definite donkey sentences, Jacobson sentences, and Bach-Peters sentences. Argument relaxation strategies are needed to handle definite donkey sentences and Bach-Peters sentences. However, the order of argument relaxation differs for the two examples. Because the relaxation strategy needed to handle Bach-Peters sentences violates the function principle, it should be a more difficult strategy for people to use.

Our functional representation of pronouns and definites is quite useful. Not only does a function model the varied behaviors of pronouns and definites, but it also limits the behaviors of those types of noun phrases. There is one caveat, however. The functional representation, though helpful, cannot be used to completely specify pronoun resolution. We do not claim that compatibility is all that must be considered to determine a pronoun's (or definite's) antecedent. Contextual information must be considered when we decide on the antecedent for a pronoun. Hence, there may be many cases when a possible antecedent for a pronoun is compatible with its function but cannot be the antecedent for the pronoun.

3.4 Past Representations of Definites

In this section, we examine previous representations of definite noun phrases. In particular, we review definite descriptions and definite quantifiers. We also examine some recent work which departs from these traditional representations of definite noun phrases (e.g., Heim [18], Roberts [41], Kamp [29], Klein [32]). Each representation is incompatible with our computational constraints, discussed in Chapter one. However, we borrowed from the insights of these approaches to design our representation for definite noun phrases.

3.4.1 Definite Descriptions

Russell [43] introduced one of the first representations for definite noun phrases. He claims that the meaning of a definite noun phrase is captured by a definite description, (ιx)(P x), which stands for *the object x such that (P x) is true*, where P is a property. This definite description names a unique object, and hence, is translated into the formula, $\exists$x (and (P x) $\forall$y (((P y)$\leftrightarrow$ (= x y))). Definite descriptions can be used to represent *the dog* in example 241 (ignoring verb phrase ellipsis).

Example 241

```
The dog: (ιx)(dog x)
The dog barked.: (barked (ιx)(dog x))
which means:
∃x (and (Dog x)                ; The dog exists.
        ∀y ((dog y) ↔ x=y)     ; It is the one-and-only dog.
        (barked x))            ; It barked.
```

Notice three important features of the meaning of the definite description in example 241. First, the dog described by the definite noun phrase is assumed to exist, as can be seen in the first line of the sentence's meaning. Second, the dog described by the definite noun phrase is assumed to be unique, as can be seen on the second line of the meaning. And finally, the definite must participate in the sentence in some way, as expressed on the third line. A definite description can be used to represent a definite noun phrase in a sentence or the quantificational meaning of a definite description can be used to represent the definite (discussed in section 3.4.3). In this section, we assume that the quantificational meaning of the definite description is provided only when the final interpretation of the sentence is available. In other words, a definite noun phrase is an in-place description in logical form, and does not participate in quantifier scope ambiguity (unless a quantified term inside of the definite description can distribute over it).

This representation suffers from several problems. First, there is no role for the effect of context on the uniqueness statement (noted by many people, including Allen [1] and Hintikka and Kulas [22]). For example, *the dog* in 241 is described as the-one-and-only *the dog*, regardless of context. Definite descriptions are fine for representing those definite noun phrases that describe unique instances regardless of context. However, *the dog* in 241 is not in this class. Second, definite descriptions do not adequately model anaphoric definites (as noted by [22]). Anaphoric definites need not be unique. Like pronouns, they seem to adapt to the behavior prescribed by their antecedents, as can be seen by example 242.

Example 242

Every boy$_i$ saw (his$_i$ dog)$_j$ before the beast$_j$ saw him$_i$.

Clearly, there is no unique animal described as *the beast* in this sentence. To cover this example, the definite description for *the beast* could be replaced by some value consistent with the representation of its antecedent, but not without violating the formal consistency constraint. Another possibility would be to devise another representation for anaphoric definites. However, this would violate our compactness constraint.

Another difficulty with in-place definite descriptions involves the representation of Bach-Peters sentences (sentences with crossing anaphoric relations). This difficulty is discussed by Hintikka and Kulas [22]. Consider the following example:

Example 230

(The boy who wrote her$_j$)$_i$ kissed (the girl who loved him$_i$)$_j$.

This sentence cannot be represented, without infinite recursion, when the definite description notation is used to provide the meaning of the sentence. To see why, consider a first stab at representing this sentence, shown in 243.

Example 243

The boy who wrote her kissed the girl who loved him.

```
(kissed (ιx)(and (boy x) (wrote x her))
        (ιy)(and (girl y) (loved y him)))
```

Before this representation captures the meaning of the sentence indicated in 230, the pronoun strings must be replaced by some value dictated by their antecedents. Since definite descriptions are not quantified terms, the pronouns cannot be replaced with variables. However, the pronouns can be replaced by the definite descriptions associated with their antecedents. Given that the antecedent for *her* is *the girl who loved him*, that pronoun is replaced by the definite description representing that definite noun phrase. This pronoun information is added to the representation shown in 243, to give the representation shown in 244.

Example 244

(The boy who wrote her)$_i$ kissed the girl who loved him$_i$.

```
(kissed (ιx)(and (boy x) (wrote x (ιy)(and (girl y)
                                           (loved y him))))
        (ιy)(and (girl y) (loved y him)))
```

Because *the boy who wrote her* is the antecedent for *him*, that pronoun is replaced by the definite description for the definite noun phrase, as shown in example 245.

Example 245

(The boy who wrote her$_j$)$_i$ kissed (the girl who loved him$_i$)$_j$.

```
(kissed (ιx)(and (boy x)
                 (wrote x (ιy)(and (girl y)
                                   (loved y him))))
        (ιy)(and (girl y)
                 (loved y (ιx)(and (boy x)
                                   (wrote x (ιy)(and (girl y)
                                                     (loved y him)))))))
```

Now as the reader can see, this process will go on infinitely[20] (as the pronoun *him* is continually replaced with the definite description for *the boy who loved her*). This recursion problem arises because of the cross-dependency between the two noun phrases, and because a pronoun whose antecedent is a definite must be replaced by the definite description representing the definite.

Despite the fact that there are a number of difficulties associated with representing definite noun phrases with in-place definite descriptions, definite descriptions have the nice properties of an in-place representation. First, they nicely capture the way that embedded pronouns affect the the meaning of a definite noun phrase when its antecedent is quantified. Consider example 246. To allow the pronoun to stand out in this example, we represent it using a pronoun function.

Example 246

Every man$_i$ loves his$_i$ mother.

```
∀x: (man x)
 x, λ(z)(love z (ιy)(and (mother y)
                         (possess (his1 z) y)
                         (= (his1 z) x)))
```

Given that *his* refers directly to *every man*, then *his mother* is represented using the definite description shown in 246. Notice that the pronoun function, (his$_1$ z), is replaced by the universal variable x. Since the quantifier binding the variable x is external to the definite description, the denotation of the definite description varies depending on the instantiation of the universal variable. Hence, once we decide that antecedent for *his* is *every man*, only the distributive reading for *his mother* is possible.

Definite descriptions can also be used to handle the ambiguous way that quantified noun phrases contained in a prepositional phrase attached to a definite affect the meaning of the definite. Consider example 116 once more.

Example 116

The head of every public authority in New York is rich.

The single head reading of *the head of every public authority in New York* is provided by keeping the quantifier associated with *every public authority in New York* inside of the definite description,

[20] Lambda abstraction of the subject won't help in this example, because the pronoun *him* would be replaced by a lambda variable in the definite description for *the girl who loved him*. This lambda variable would become unbound when the description for *the girl who loved him* replaces the pronoun *her* in the definite description for *the boy who wrote her*.

as shown in example 247.

Example 247

```
The head of every public authority in New York is rich.

(ιx)(head-of x [∀y: (and (public authority y)
                         (in y New York)) y]), λ(z)(rich z)
```

On the other hand, the distributive head reading is provided by moving the quantifier outside of the definite description, as shown in 248.

Example 248

```
The head of every public authority in New York is rich.

∀y: (and (public authority y) (in y New York))
 (ιx)(head-of x y), λ(z)(rich z)
```

Despite the fact that definite descriptions can express the ambiguity of the definite noun phrase in 116, we still need to determine whether the embedded quantifier should distribute over the definite. Additionally, we must be able to determine prepositional phrase attachment, if an ambiguity arises.

Finally, definite descriptions are a nice representation for verb phrase ellipsis (especially when compared with the quantified representation we discuss in the next section). Consider example 6.

Example 6

```
Trigger Sentence: Fred_i loves his_i wife.
Elided Sentence:  George_j does too.
Meanings:
1. George loves Fred's wife.
2. George loves George's wife.
```

The trigger sentence is represented in two ways, as shown in 249.

Example 249

```
Fred_i loves his_i wife.

a. (ιx)(name x Fred), λ(y)(love y (ιz)(and (wife z)
                                           (possess y z)))
b. (ιx)(name x Fred), λ(y)(love y (ιz)(and (wife z)
                                           (possess (ιx)(name x Fred) z)))
```

Because the subject is the antecedent for *his*, the pronoun can either be replaced by the lambda variable y or by the definite description associated with the subject. The first representation of the trigger sentence, shown in 249a, is derived by replacing the pronoun with the subject's lambda variable. The second representation, shown in 249b, is derived by replacing the pronoun with the definite description for the subject. Once two representations of the trigger sentence are available, the two readings for the elided sentence are derived, as shown in 250.

Example 250

George does too.

```
a. (ιw)(name w George), λ(y)(love y (ιz)(and (wife z)
                                              (possess y z)))
   ; George loves George's wife.
b. (ιw)(name w George), λ(y)(love y (ιz)(and (wife z)
                                              (possess (ιx)(name x Fred) z)))
   ; George loves Fred's wife.
```

Because (ιx)(name x Fred) contains no free variables in 249b and 250b, it denotes the same individual in both (because of uniqueness). Hence, the strict reading of the elided sentence can be derived (unlike the definite quantifier representation discussed in Chapter two, section 2.2.2). The sloppy reading of the elided sentence is also available, as shown in 250a. Definite descriptions can also be used to handle example 80 (which is a problem for many models of verb phrase ellipsis). Crossing reference sentences, however, are a problem for definite descriptions, as we have already shown.

If we create a model of language comprehension using in-place definite descriptions to model definite noun phrases, the model would not satisfy our computational constraints, described in Chapter one. When definite descriptions are used to model definite anaphora, the formal consistency constraint can be violated. Because of uniqueness, replacing a definite description with some value corresponding to its antecedent can cause a violation of the formal consistency constraint. To avoid this problem, two different representations for a definite could be provided (one new representation for anaphoric definites plus definite descriptions), but this solution violates the compactness constraint. There is no way to represent both the anaphoric and non-anaphoric readings of a definite noun phrase using definite descriptions without violating either compactness or formal consistency. However, the definite description for a definite noun phrase can be provided without violating modularity (assuming that attachment ambiguities are resolved before the representation is provided).

The assets of in-place definite descriptions guided our functional representation of definite noun phrases. However, since there is no reason short of traditional and historical reasons for using a definite description, we claim that our functional representation of a definite noun phrase should be used to model definite noun phrases. In particular, our representation easily covers the behaviors of both anaphoric definite noun phrases and definite noun phrases with complex structure. Our representation does not have the infinite recursion problem that Bach-Peters sentences present for definite descriptions, and yet we have maintained many of the nice properties of definite descriptions. For example, we can handle the distributive aspect of a definite, as well as the ambiguity that arises when a quantified noun phrase is contained in a prepositional phrase attached to a definite. Finally, we do not have the problem with uniqueness which definite descriptions have (because every definite function has a unique name). Different functions with the same restriction do not have to denote the same individual, though they could. This seems to be a better approach than one that assumes that two things described with the same words must denote the same object.

3.4.2 Definite Descriptions in Webber (1978)

Webber [49] represents definite noun phrases as definite descriptions. She chooses the definite operator instead of a quantificational representation for a definite. She indicates no reason for her choice, but does indicate that while the choice does not affect what can be represented in her system, it does affect the form of her rules for generating discourse entities or trigger verb phrases. While a quantifier scopes an open sentence, an in-place definite description simply fills an argument slot in a predicate-argument structure representing the sentence.

Despite the fact that Webber represents definite noun phrases as definite descriptions in the logical form for a sentence, she must provide some way to circumvent their problems. To overcome the fact that definite descriptions do not cover anaphoric definite noun phrases, Webber distinguishes between two levels of representation for a sentence. In the level-1 representation of a sentence, there is no difference between those definite noun phrases used anaphorically and those used to evoke discourse entities. All definite noun phrases are initially represented as definite descriptions. The transition from a level-1 to a level-2 representation for a sentence requires that each definite description be examined to determine whether or not it is used anaphorically. If it is, then the definite description is replaced by the value associated with its antecedent, otherwise, it is left alone. However, it may not be possible to determine whether or not a definite noun phrase is used anaphorically (e.g., there may be unresolved pronouns in the definite) during the transition between the two levels of representation. Additionally, the replacement of a definite description by some value corresponding to its antecedent can violate the formal consistency constraint.

Webber's model handles the Bach-Peters sentence in example 230 by allowing the pronouns to be replaced by the discourse entities of their definite antecedents. However, her model cannot handle Jacobson sentences. In fact, the model fails in a way which is quite similar to its failure with example 76 (discussed in section 2.4.2).

Example 235

Everyone$_i$ told her$_j$ mother$_k$ that (his$_i$ wife)$_j$ should get a job.

Webber's level-1 representation for this sentence is shown below:

Example 251

```
Every man told her mother that his wife should get a job.
∀x: (person x)
 x, λ(y) (tell (ιz)(mother-of (her))
              [(ιw)(wife-of (his)), λ(s)(∃t: (job t) (get s t))])
```

Now given that the antecedent for *his* is *everyone* and the antecedent for *her* is *his wife*, the model cannot provide the expected meaning for the sentence. The pronoun *his* can be replaced by the lambda variable y or by the universal variable x. However, when the antecedent for *her* is *his mother*, Webber's model replaces the pronoun with a discourse entity. This, however, does not capture the meaning of this sentence. Webber's model could capture the meaning of the sentence by replacing the pronoun *her* by the definite description for *his wife*. However, this would reinstate the difficulties with Bach-Peters sentences that Webber avoids (by using the idea that pronouns should be replaced by discourse entities, not definite descriptions).

In summary, Webber's model avoids many of the difficulties that the use of definite descriptions would normally introduce (some of those problems have been reviewed in section 3.4.1). However, to

avoid those difficulties, the model introduces a new set of problems. In particular, the assumption, pronouns with non-subject definite antecedents are replaced by discourse entities, causes problems. Additionally, Webber's model violates our computational constraints. For example, Webber's initial representation of an anaphoric definite is modified between levels one and two of her model. When a definite is used anaphorically, the model replaces the definite description (used to initially represent the definite noun phrase) with some value consistent with its antecedent. This modification can violate the formal consistency constraint.

3.4.3 Definite Quantification

Other researchers have represented definites using the quantificational meaning of a definite description directly in the representation of a sentence (e.g., Webber [50] and Montague [35]). For example, the sentence, *The dog barked*, could be represented as shown in 252.

Example 252

`The dog barked.`

$\exists$x: (and (dog x) ($\forall$y ((dog y) $\leftrightarrow$ x=y))) (barked x)

A short-hand notation for this is indicated below:

Example 253

`The dog barked.`

$\exists!$x: (dog x) (barked x)

The quantifier $\exists!$ means there exists a unique. The question that we must ask is, does the use of this quantifier improve our ability to model the behavior of a definite noun phrase? We explore the effectiveness of such an approach in this section.

The quantificational representation for definites suffers from several problems. First, as one might guess, it suffers from the same uniqueness problem that in-place definite descriptions have. For example, the noun phrase *the dog* (in example 252) is represented with the assumption that it is the-one-and-only dog. This problem must be fixed for definite quantifiers to provide a reasonable representation for definite noun phrases. Dowty, Wall, and Peters [16] suggest a nice way to fix the uniqueness problem. To eliminate the one-and-only aspect of a definite, they relativize uniqueness to a context of utterance (much as the domain of a universal noun phrase must be relativized to a context of utterance). The precise details of this measure were not elaborated; however, they did suggest that definites should be evaluated with respect to some index dependent on the definite's location in discourse. Though this solution improves definite quantifiers, it does not eliminate their problem with respect to definite anaphora. Definite quantifiers must violate either compactness or formal consistency to model anaphoric definites. Dowty, Wall, and Peter's repair for uniqueness does not provide a way to handle sentences like those in examples 225 or 150.

Another problem with this representation is the fact that other noun phrases represented using quantifiers exhibit behaviors that definite noun phrases do not share. For example, negation seems to affect quantified noun phrases but not definites[21]. Negation does not affect the uniqueness part

[21] The only situations where one might make the argument that negation affects a definite noun phrase arises with

of the definite. Because of this, Dowty, Wall, and Peters [16] suggested that uniqueness should be a conversational implicature rather than an entailment of the quantifier. To see why consider example 254.

Example 254

The dog didn't bark.

If negation has scope over the definite, then it should be possible to get the interpretation that *There is no unique dog that barked.* However, this sentence can only mean *There is some specific dog that didn't bark.*

Quantified noun phrases often participate in quantifier scope ambiguities, many of which do not affect definites. For example, compare ***Every man loves a woman*** with ***Every man loves the woman.*** There are two meanings for the first sentence, as shown in 255.

Example 255

```
Every man loves a woman.

a. ∀x: (man x)  ∃y: (woman y) (loves x y)
b. ∃y: (woman y) ∀x: (man x) (loves x y)
```

As one might expect, there are also two representations for the other sentence, given a quantificational representation for definite noun phrases, as shown below:

Example 256

```
Every man loves the woman.

a. ∀x: (man x) ∃!y: (woman y) (loves x y)
b. ∃!y: (woman y) ∀x: (man x) (loves x y)
```

However, while the two representations in 255 express different meanings, the representations in 256 express the same meaning. The two representations in 256 express the same meaning, because uniqueness ensures that they are equivalent. In order for 256a to provide a different reading from 256b, the variable x would have to be found in the restriction of the quantifier $\exists!y$[22]. Despite the fact that ambiguity does not arise in 256, definite scope ambiguity does arise under certain circumstances. For example, consider the first sentence in 257.

---

sentences like *I did not have lunch with the king of France.* Russell argues that this sentence is true (in our world) only when the negation has *scope* over the definite (hence the sentence asserts that there is no king of France). The sentence is false if the negation does not have scope over the definite (because in this situation the sentence is asserting that the king of France exists). However, is existence in the real world always implied by the use of a definite? If so, are sentences false when the thing described by the definite fails to exist? Are they ill-formed? These are interesting questions, though not central to this thesis. On the other hand, we are quite interested in whether definites act like quantifiers with respect to their intersentential behavior (e.g., can a definite noun phrase be the antecedent for a pronoun in another sentence?

[22] To determine whether these two representations of the sentence are equivalent requires additional processing in this approach.

Example 257

In each car, the mechanic adjusted the steering wheel.
He found that they were all binding.

Notice, the universal *each car* can have scope over *the steering wheel*, giving a distributive reading.

Webber, who originally handled definites using definite descriptions [49], later chose to represent definite noun phrases by using definite quantifiers [50]. One reason she chose definite quantifiers was because of the ambiguity found in 257. She observed that definites can have same-per and different-per readings, just like indefinites (when there is a universal that could have scope over it). Notice that, if the universal *each car* has scope over the definite *the steering wheel*, then the different-per reading is available. If the universal does not have scope, then the same-per reading is available. However, this definite scope ambiguity arises only because the quantified noun phrase *each car* is the object of the prepositional phrase attached to the definite noun phrase *the steering wheel*. Definite scope ambiguity seems to arise only when a prepositional phrase containing a quantified noun phrase is attached to a definite noun phrase, not in other cases. However, a quantifier representation for definites always provides multiple representations when there are other quantifiers in the sentence, as we showed in example 256. Definite quantifiers seem to violate a constraint more fundamental than our computational constraints, that is to provide multiple representations for a sentence only when an ambiguity is present. By using definite descriptions or functional representations for definites, we avoid this problem. Why resort to multiple representations for the same meaning when ambiguity arises only when a quantified noun phrase is contained in a prepositional phrase attached to a definite, especially when there are a number of other problems associated with representing definites as quantified terms.

Finally, pronoun references to definites are not constrained in the way that pronoun references to other quantified noun phrases are. For example, when a quantifier is embedded in a relative clause attached to a noun phrase, it cannot bind a pronoun in the matrix clause, as shown by the sentence in example 39.

Example 39

*Fred$_i$ gave the psychiatrist who cares for every woman$_j$ her$_j$ diary.

In this case, *every woman* cannot be the antecedent for *her*. In contrast, a pronoun can refer to a definite contained in a relative clause attached to a noun phrase (so long as compatibility can be established), as can be seen in 38.

Example 38

Every man$_i$ gave the psychiatrist who cares for (his$_i$ mother)$_j$ her$_j$ diary.

There is a similar problem with the Bach-Peters sentences. Consider the sentence shown in 230.

Example 230

(The boy who wrote her$_j$)$_i$ kissed (the girl who loved him$_i$)$_j$.

In this crossing-reference sentence, when both of the meanings of the definites can be expressed as constants, the sentence can receive the meaning indicated given the indices on the noun phrases. However, the cross-dependency is never allowed if the noun phrases are universals as opposed to definites, as shown in 258.

Example 258

*(Every boy who wrote her$_j$)$_i$ kissed (every girl who loved him$_i$)$_j$.

The antecedent for *her* cannot be the quantified noun phrase in the object position of the sentence in 258. A similar problem arises for the Jacobson sentences. Consider the sentence shown in 235.

Example 235

Everyone$_i$ told (her$_j$ mother)$_k$ that (his$_i$ wife)$_j$ should get a job.

While the co-indexing is fine for the sentence in 235, when the definites are replaced by universals, then the sentence cannot receive the reading indicated by the co-indexing of noun phrases (see 259).

Example 259

*Everyone$_i$ told (every woman who knows her$_k$)$_j$ that (every woman who knows him$_i$)$_k$ should get a job.

If definites and universals are both quantified, then why is it that definites behave so differently from universals?

One might, like Hornstein [27], decide that definites are quantified but act very differently than universal quantifiers. Hornstein contrasts type one and type two quantifiers. The prototypical type one quantifier is a proper noun; whereas, the prototypical type two quantifier is a universal. Type two quantifiers behave in all of the ways that linguists typically think that quantifiers behave. In contrast, type one quantifiers, which include definites, act differently. Hornstein contrasts type one quantifiers with type two quantifiers on many features. He claims that while type two quantifiers are more restricted in their ability to bind pronouns (i.e., they cannot bind a pronoun in another sentence), type one quantifiers can bind more widely (i.e., across or within sentences). He claims that while the leftness constraint (i.e., pronoun cannot be co-indexed with a type two quantified noun phrase to its right [10,19]) holds for type two quantifiers, it does not hold for type one quantifiers. While a type two quantifier is sensitive to its logical environment, a type one quantifier is usually insensitive to its logical environment. Hence, a type one quantifier seems to act as if it has widest scope or like a name. Finally, type two, but not type one, quantifiers are subject to locality restrictions on movement like the Empty Category Principal[23].

The problem with this approach is that there is no discussion about when a type one quantifier should be sensitive to its logical environment. The assumption that definites are like names doesn't always hold. Definites are affected logically by embedded quantified terms not subject to island constraints and the fact that embedded pronouns can be bound by type two quantifiers. Additionally, there is no framework to specify just how type one quantifiers do work. Why is it that a definite containing a pronoun can be the antecedent for a singular pronoun in another sentence when the pronoun has a certain antecedent but not another? Definite binding must be very complicated, resting particularly on which antecedents are chosen for pronouns embedded in definite noun phrases. Hornstein does not specify when definite binding is possible. Furthermore, the approach does not correct some other problems of definite quantifiers, like uniqueness or the inability to

[23]The Empty Category Principal is a well-formedness constraint on quantifier movement in Government Binding Theory.

handle anaphoric definites.

Rather than attempting to patch up definite quantifiers, it seems far better to represent definites as functions. First, the representation of a definite noun phrase as a function provides us with a way to handle anaphoric definite noun phrases, in contrast to definite quantifiers. Additionally, we represent definites as a function with a restriction. The restriction provides us with a nice way to determine what the arguments of the definite function should be, depending on which variables are bound by operators located outside of the restriction of the definite function. Hence, we are able to capture the nice properties of a definite description while not neglecting anaphoric definites. Additionally, we do not have the problem with uniqueness which both definite descriptions and definite quantifiers share. Finally, while definite quantifiers can violate the compactness and formal consistency constraints (formal consistency is violated when a wide scope quantified term is replaced by something that cannot be described as a constant and compactness is violated when a different representation is introduced to handle anaphoric definites), our approach obeys our constraints more closely.

3.4.4 Heim (1982)

Heim [18] does not represent definite noun phrases using definite descriptions or quantifiers. In fact, in her dissertation, she spent much time and effort trying to eliminate the traditional existential quantification representation for an indefinite. Heim claims that definites and indefinites should be treated very similarly since both can be referred to across sentence boundaries, unlike universal noun phrases. She distinguishes definites from indefinites using felicity conditions of speech. In particular, she claims indefinites always introduce a novel referent, not already mentioned in discourse. On the other hand, definites are used infelicitously if they do not refer to some referent already mentioned in discourse. This felicity condition is called the novelty-familiarity condition on speech.

Heim builds a discourse model for language with three levels, interpretation rules (which result in an augmented parse tree, that is a parse tree with quantifier scoping information and pronoun antecedents specified), felicity conditions (e.g., the novelty-familiarity condition, introduced to filter out inappropriate usage of definites and indefinites), and accommodation rules (which provides a way to introduce a discourse referent for a definite by linking it to some previous discourse referent, an exception to the familiarity condition on definites). Heim represents a definite noun phrase by using file change semantics (we will illustrate the use file change semantics with an example shortly) which is limited by the parse tree, felicity conditions, and accommodation. Heim's initial representation of the meaning of a sentence requires pragmatic information, not just syntactic information, as we will show.

To provide an interpretation for a sentence Heim determines the *logical form* for a sentence. The *logical form* is essentially a parse tree with quantifier scoping information indicated in it, but it is not a logical representation for the meaning of the sentence. For example, in Heim's *logical form*, definite noun phrases are represented as a subtree with leaf nodes corresponding to the words in the noun phrase. Once the *logical form* for a sentence is constructed, she provides a file change semantics for the sentence (using felicity conditions and accommodation to limit the possible meanings the sentence can have). One of the most important felicity conditions for constructing a reasonable discourse representation for a series of sentences is the novelty-familiarity condition. This condition states that indefinites should always cause a new discourse referent (or file card) to be created in the discourse model. On the other hand, definite noun phrases should not introduce a new discourse referent (or file card). She provides a way to construct a discourse model from the

logical forms for a set of sentences. We will not show the *logical forms* here, but we will discuss the process of constructing the discourse model for a series of sentences.

Consider an example Heim discusses to demonstrate the use of file change semantics.

Example 260

(a) A woman was bitten by a dog.
(b) She hit him with a paddle.
(c) It broke in half.
(d) The dog ran away.

At the beginning of processing, the file for the listener is empty. After (a) is uttered, two new file cards are introduced. The statement *is a woman* is written on card 1, and *is a dog* is written on card 2 . To add sentential information, she adds *was bitten by 2* to card 1 and *bit 1* to card 2. Because Heim claims that indefinites always introduce new file cards, this addition to the model is consistent with her felicity conditions. After (b) is uttered, one new file card is created. The statement *is a paddle* is written on card 3. Since Heim classifies pronouns as definites, they must always cause an existing card to be updated. Because *a woman* is the antecedent for *she*, Heim updates the first file card with information pertaining to the pronoun *she*. Also, assuming that the antecedent for *him* is *a dog*, then she updates file card 2 with information pertaining to *him*. To add sentential information, she adds *hit 2 with 3* to card 1, *was hit by 1 with 3* to card 2, and *used by 1 to hit 2* to card 3. After (c) is uttered, she simply updates card 3 to indicate sentential information by adding *broke in half* to card 3 (because the antecedent for *it* is the discourse referent represented with file card 3). Finally, after (d) is uttered, she adds *ran away* to card 2 (because the antecedent of *the dog* is represented as file card 2). In keeping with her felicity conditions, a new card is introduced for every indefinite. Additionally, for every definite, an old card is updated.

Clearly Heim's approach emphasizes the anaphoric aspect of the definite noun phrase, which both definite descriptions and definite quantifiers fail to handle well. In fact, Heim's model handles anaphoric definites, but possibly at the cost of not handling definites with complex structure well. Consider example 261.

Example 261

Every man$_i$ loves his$_i$ mother.

In order for Heim to handle examples like this, she relies on the process of accommodation. Accommodation is a rather ill-defined process (originally introduced by Lewis [33]) which allows Heim to introduce a new file card for a definite noun phrase if and only if that noun phrase is related to a previous file card. Given the indexing on the noun phrases, *his mother* is non-anaphoric. However, given that the antecedent for *his* is *every man*, file change semantics handles this example by virtue of accommodation. Since there is a file card for *every man*, the pronoun *his* provides the needed link for accommodation to go through. In order to provide a file card for *his mother*, Heim must know that *his* refers to *every man*. Hence, accommodation seems to require more than syntax and sentence-level information, pragmatic information is necessary to provide the representation for definite noun phrases, in violation of the modularity constraint.

Consider another accommodation example, discussed by Heim [18].

Example 262

John read a book about Schubert and wrote to the author.

In order for Heim to represent the meaning of this sentence using file change semantics, she must bridge the gap between *a book about Schubert* and *the author*, given that *the author* does not refer to some previously mentioned author. In this case, accommodation is needed to introduce a new file card for *the author*. For accommodation to succeed, the inference that *the author* fills the author slot in the *a book about Schubert* (which has already been given a file card representation) is necessary. Hence, to handle the sentence in 262, inference and world knowledge is required. In Heim's approach, the representation of a definite noun phrase often requires more than syntactic or sentence-level information, thus violating the modularity constraint (i.e., the initial representation of a definite noun phrase must be available based only on syntactic and sentence-level information). We certainly want to bridge this gap between the two noun phrases, but not initially. This step should be handled later in processing.

Accommodation, as Heim points out herself, is not a fully developed concept.

> Under which conditions is accommodation an available option, and what exactly is added to the file when this option is taken? These questions are by no means easy to answer, as can be seen from the attention that they (or rather the analogous questions that correspond to them in other theoretical frameworks) have received in the literature, ranging from the work of traditional grammarians to much recent work in psychology and artificial intelligence. From the point of view of much of that work, they are perhaps the only non-trivial questions that a theory of definiteness faces. Be that as it may, I can say only very little about the rules that govern accommodation, none of which is new. (p. 372)

Hence, accommodation is a little understood process on which Heim builds half of her model of definite noun phrases. That is, she uses it to explain the behavior of non-anaphoric definites. Accommodation may be a psychologically motivated process for sentence comprehension. However, accommodation should be used later in processing.

There is another problem with using accommodation to represent non-anaphoric definite noun phrases. How does Heim's model handle definite noun phrases that are non-anaphoric and have no accommodation link to a previous noun phrase in the discourse model? For example:

Example 263

Fred saw the building of every architect in the firm.

The definite noun phrase *the building of every architect in the firm* has a very complex structure, no link to a noun phrase outside of those embedded in it, and it is probably used non-anaphorically. In order for Heim to handle this example, she might introduce a file card for *every architect in the firm* and then accommodate *the building of every architect in the firm* by using that card[24]. However, she must distinguish between the single-building reading and the distributive reading of the noun phrase. Certainly the distributive reading is naturally handled by introducing the file card for *every architect in the firm* first. However, the definite noun phrase can also denote a single building, and that building may not have been mentioned previously in discourse. Under these

[24]Heim [18] does not discuss examples like 263.

circumstances, there is no link to a previously introduced file card and no way to introduce a new file card without violating the novelty-familiarity constraint.

Heim's approach requires a considerable amount of information to be known before a definite noun phrase is represented in file change semantics. Her *logical form* for a sentence is not a logical representation for the meaning of the sentence, though she does indicate quantifier scoping information on the tree. Before the definite noun phrase is provided a semantic representation, a considerable amount of pragmatic information has been used to provide a single *logical form* for the sentence (in violation of modularity). Once the *logical form* is available, Heim maps it into her discourse model using file change semantics. Indefinites cause the introduction of a new file card in the discourse model. Definites usually do not, unless there is some link to a previous file card that the process of accommodation can use. Before a file card can be introduced for a non-anaphoric definite, pronoun antecedents have to be known, providing another violation of the modularity constraint. If this information is not available, then compactness is also violated (since depending on the antecedents chosen for pronouns or the quantifier scoping chosen for a sentence, different representations of a sentence will be provided in file change semantics). The process of representing sentences in file change semantics does not seem to be consistent with our goal to incrementally determine the meaning of a sentence. There is no sense of how to divide the process of sentence comprehension into small, independent steps which eventually allow us to reach the goal of an unambiguous meaning for the sentence.

In our approach, we represent both anaphoric definites and definites with a complex structure as functions. We do not need to know quantifier scope information, pronoun antecedents, or accommodation links to provide an initial representation for a definite noun phrase (even when it is non-anaphoric). Because we represent definite noun phrases as functions of all of the variables associated with operators that can affect them, we can update those functions to handle the anaphoric case, and we can also handle non-anaphoric, distributive definites. Because we use minimal information to provide the initial representation, our approach is more amenable to implementation.

3.4.5 Discourse Representation Theory

Kamp [29] introduces a discourse model theory similar to Heim's, called Discourse Representation Theory. In particular, both theories are motivated by the fact that pronouns in one sentence can have definite and indefinite antecedents in another sentence, while universals cannot bind pronouns in other sentences. Kamp's approach has been extended by several researchers (e.g., Klein [32], Roberts [41]). We introduce Discourse Representation Theory, not as it was given in Kamp [29], but as it is discussed in Roberts [41]. Because Kamp does not discuss quantifier scope ambiguities or definites, a later view of Discourse Representation Theory provides a more comprehensive theory to examine. One nice feature of Discourse Representation Theory is the fact that the discourse models created have model-theoretic interpretations.

In Discourse Representation Theory, a set of construction rules converts natural language into discourse structures. To do so, however, quantifier scoping information must be specified, since the discourse model provided depends on the quantifier scoping chosen for the sentence. Consider how example 264 is handled in Discourse Representation Theory.

Example 264

```
Someone loves everyone.
```

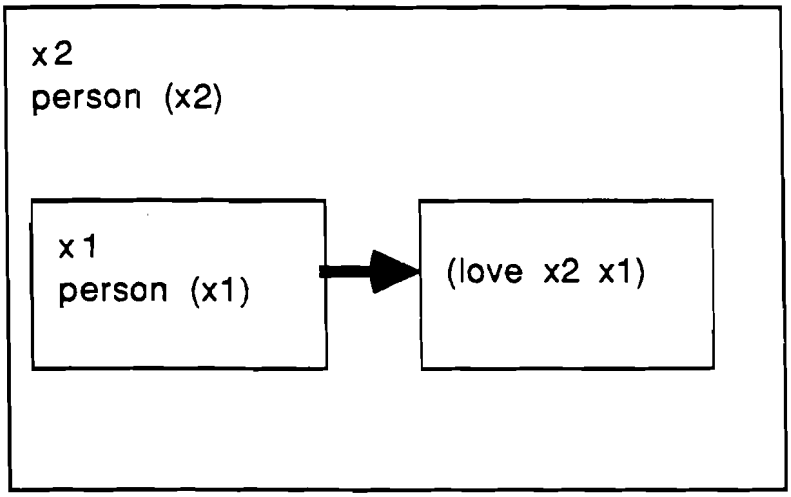

Figure 3.2: Discourse Representation of Example 264: *someone* has scope over *everyone*

This sentence has two possible meanings (corresponding to the two quantifier scoping orders). In one case, *someone* has scope over *everyone*. Hence, *someone* acts like a constant. This representation is shown in Figure 3.2. In this case, the discourse referent for *someone* is *x2* and the discourse referent for *everyone* is *x1*. Universal noun phrases always introduce an antecedent-consequent box like the one shown in Figure 3.2. Because *x2* is defined outside of the influence of *x1* in this representation of the sentence, it acts like a constant. In contrast, consider how the sentence is represented if *everyone* has scope over *someone* (shown in Figure 3.3). In this case, because the discourse referent for *someone* is created in the consequent box of the universal, its denotation depends on *x1*, the discourse referent for *everyone*. Hence, each quantifier scoping requires a different discourse representation for the meaning of the sentence.

In Discourse Representation Theory, mapping into a discourse representation is a top-down process, reducing the original sentence to a structure with a discourse referent for each noun phrase, with predicates indicating restrictions on the discourse referents, and with predicates indicating relations between discourse referents. As we already pointed out, universals are represented by placing their discourse referents into an antecedent box, with additional sentence information placed in the consequent box (giving a meaning very much like a universal in predicate calculus). Indefinites and definites are represented by placing their discourse referents and restriction information in the box corresponding to the current level in the model. An accessibility relation determines when a pronoun can have a particular discourse referent as its antecedent. A pronoun's antecedent can be any discourse referent defined in the box where the pronoun is instantiated or in any box containing that box. Additionally, a pronoun in a consequent box can also refer to anything in the antecedent box (unless the antecedent is embedded in another box contained in the antecedent box). This accessibility relation is quite useful for handling donkey sentences, as we will show soon.

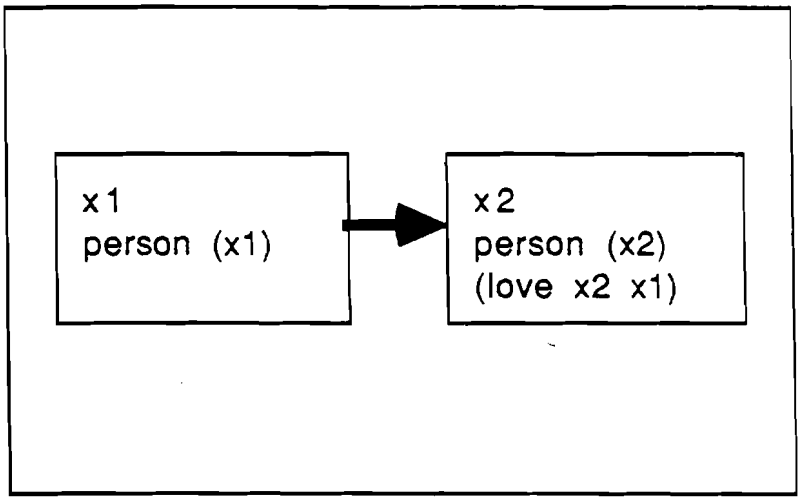

Figure 3.3: Discourse Representation of Example 264: *everyone* has scope over *someone*

In [41], Roberts combines Discourse Representation Theory with the notion of c-command to handle the meanings of sentences. She claims that because pronouns need an antecedent, they act like variables waiting to be bound by their antecedents. She distinguishes two types of binding, c-command binding and discourse binding. C-command binding occurs when the best way to represent the anaphoric noun phrase is by replacing it with the variable associated with the operator of the noun phrase that c-commands it. On the other hand, discourse binding is needed to handle anaphoric dependencies on things that don't c-command a pronoun or definite noun phrase. For example, consider the sentence in 265.

Example 265

```
Every miner who owns a donkey beats it.
```

If *a donkey* had c-commanded *it*, then no discourse referent would be created for *it*. Instead, the pronoun would be represented using the discourse referent of its antecedent. However, because *a donkey* does not c-command *it*, there is no way for a c-command binding to occur. The sentence is handled in Discourse Representation Theory, as shown in Figure 3.4. Notice that the pronoun is represented as a discourse referent *x3*, which is equated with the discourse referent for *a donkey* (i.e., *x2*). Pronouns that haven't already been replaced by a discourse referent, must be equated with some accessible discourse referent. As we already said, pronouns can only accept antecedents in their own box, in a higher box, or in an antecedent box. Thus, discourse binding provides a way to handle the donkey sentence (as well as example 217).

Roberts also attempts to handle verb phrase ellipsis in Discourse Representation Theory. However, because Klein's [32] model of verb phrase ellipsis is better than Robert's model, we concentrate on Klein's approach. To handle verb phrase ellipsis in Discourse Representation Theory, Klein introduces a device which is very similar to lambda abstraction. With this device, he is able to

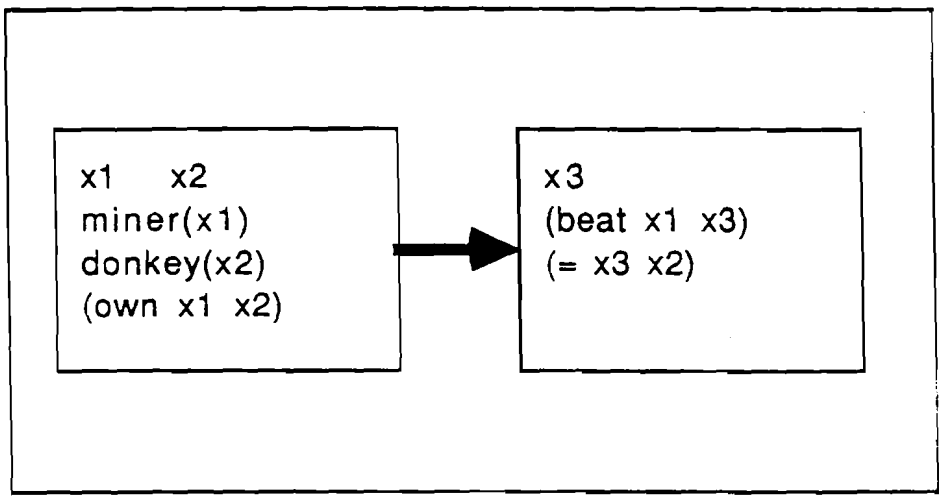

Figure 3.4: Discourse Representation of the donkey sentence in Example 265

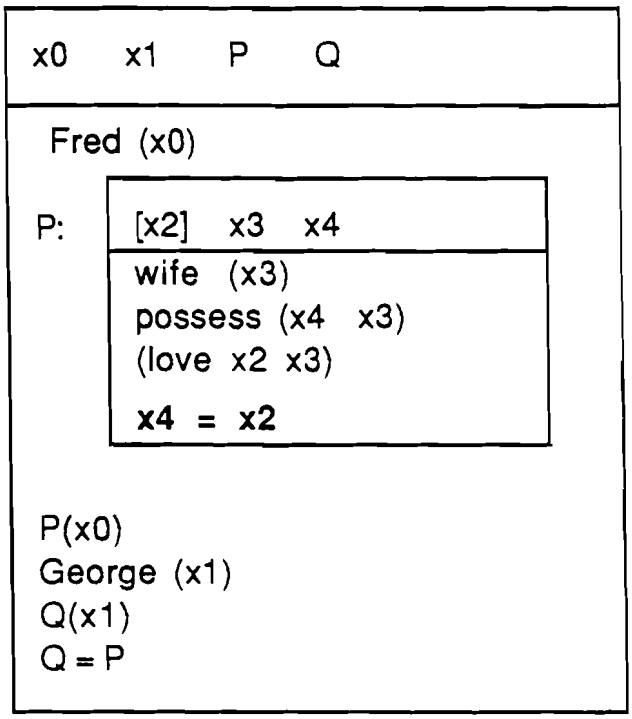

Figure 3.5: A Discourse Representation for the sloppy reading of Example 6

represent the verb phrase and *abstract* the syntactic subject of the sentence. Consider how he handles example 6.

Example 6

```
Trigger Sentence: Fred_i loves his_i wife.
Elided Sentence:  George_j does too.
Meanings:
1. George loves Fred's wife.
2. George loves George's wife.
```

The discourse representation for the sloppy reading of the elided sentence is shown in Figure 3.5. Klein represents the verb phrase for the trigger sentence as a boxed structure named P. Within this box is a distinguished variable *x2* (distinguished variables are marked with brackets), which corresponds to the abstracted subject. The trigger sentence is represented as *P(x0)*, which is very similar to applying the discourse referent for the subject to a lambda function named P. Now the discourse referent for *his* is *x4*, which can either be equated with the distinguished discourse referent (i.e., *x2*) or with something outside of the verb phrase box. To get the sloppy reading, it is equated with the distinguished discourse referent, as shown in Figure 3.5. The elided sentence is represented initially as *Q(x1)*, where *x1* is the discourse referent for the subject of the elided sentence. The sloppy reading for the sentence is provided when *Q* is equated with *P*. On the other hand, to derive the strict reading of the elided sentence, the discourse referent for the pronoun in the verb phrase is equated with the discourse referent for the subject, namely *x0* (as shown in

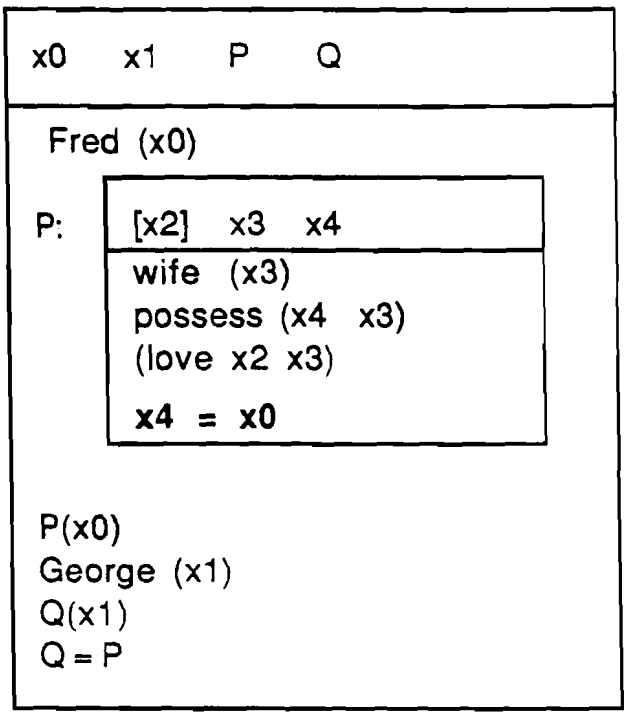

Figure 3.6: A Discourse Representation for the strict reading of Example 6

Figure 3.6). Again the meaning of the elided sentence is derived by equating Q with P, but in this case the contents of P are slightly different (i.e., the pronoun is equated with the discourse referent for the subject). Thus, the strict reading of the elided sentence is also provided. Hence, Klein derives the two expected readings for the elided sentence in 6 by introducing lambda abstraction to Discourse Representation Theory.

This approach to verb phrase ellipsis behaves quite similarly to our approach to verb phrase ellipsis. However, there is a question about how to deal with definites, since they are not discussed in detail. To see why this is an issue compare examples 266 and 267.

Example 266

```
Fred saw a dog.
George did too.
Meanings:
1. George saw the same dog that Fred saw.
2. George saw a different dog than Fred saw.
```

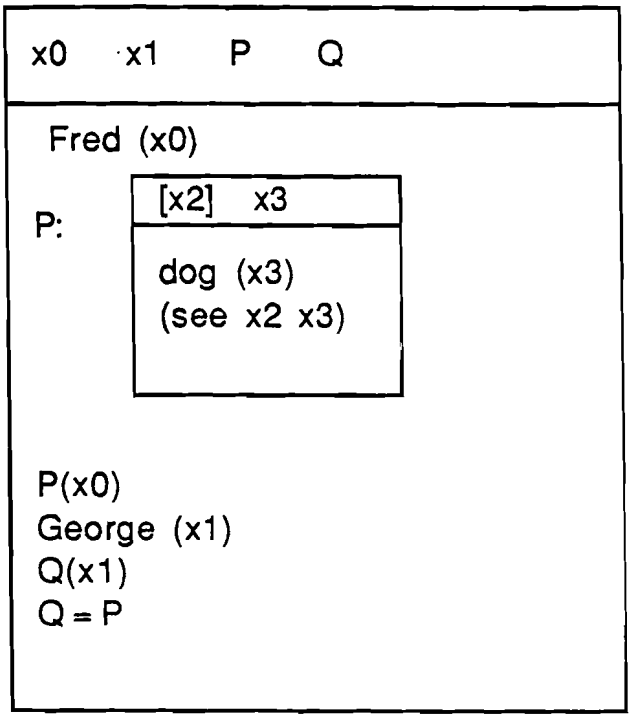

Figure 3.7: A Discourse Representation for Examples 266 and 267

Example 267

```
Fred saw the dog.
George did too.
Meanings:
George saw the same dog that Fred saw.
```

These two examples would have precisely the same discourse representation, shown in Figure 3.7. However, in verb phrase ellipsis, the indefinite case is clearly different than the definite case (i.e., the indefinite introduces an ambiguity that the definite does not share). Klein could use Heim's felicity conditions on definites in order to distinguish the two uses (something which Roberts suggests in her dissertation). However, as we have already discussed, these conditions only hold when a definite is anaphoric, otherwise accommodation is needed.

The question is, is there some basic difference between definites and indefinites independent of the anaphora issue? We believe that there is. It is possible to begin a story with a sentence like *The girl tripped over the dog.* Under these circumstances, there is no antecedent for either noun phrase. However, if the second sentence is *The boy did too*, it is a necessary fact that *the boy* tripped over the same dog as *the girl* did. Non-anaphoric definites without embedded pronouns are necessarily constants.

The lack of difference between definites and indefinites presents other problems for Klein's approach. Consider examples 268 and 269.

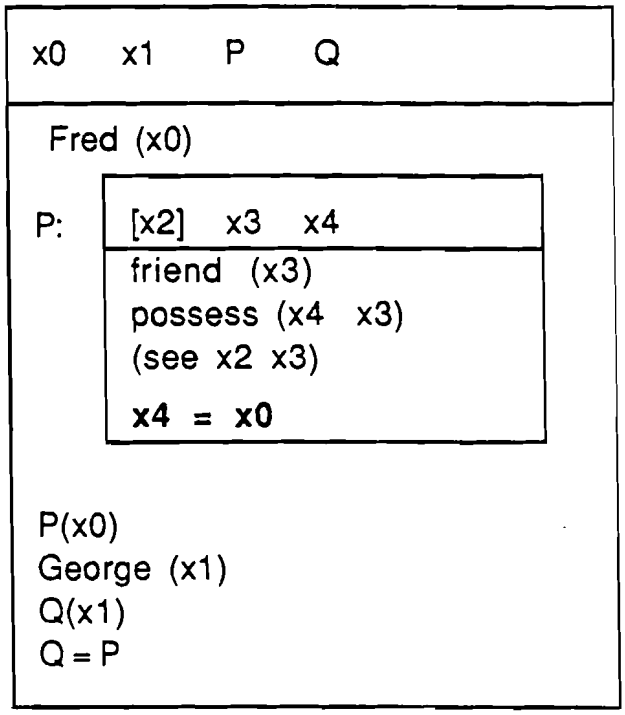

Figure 3.8: A Discourse Representation for Examples 268 and 269: *his* refers directly to *Fred*

Example 268

Fred$_i$ saw a friend of his$_i$.
George$_j$ did too.
Meanings:
1. George saw the same friend of Fred's.
2. George saw a different friend of Fred's.
3. George saw a friend of George's.

Example 269

Fred$_i$ saw his$_i$ friend.
George$_j$ did too.
Meanings:
George saw Fred's friend.
George saw George's friend.

The strict readings for both of these examples are represented in precisely the same way (shown in figure 3.8). The sloppy readings are also represented in the same way (shown in figure 3.9). However, the meaning of *his friend* is quite different from the meaning of *a friend of his* in verb

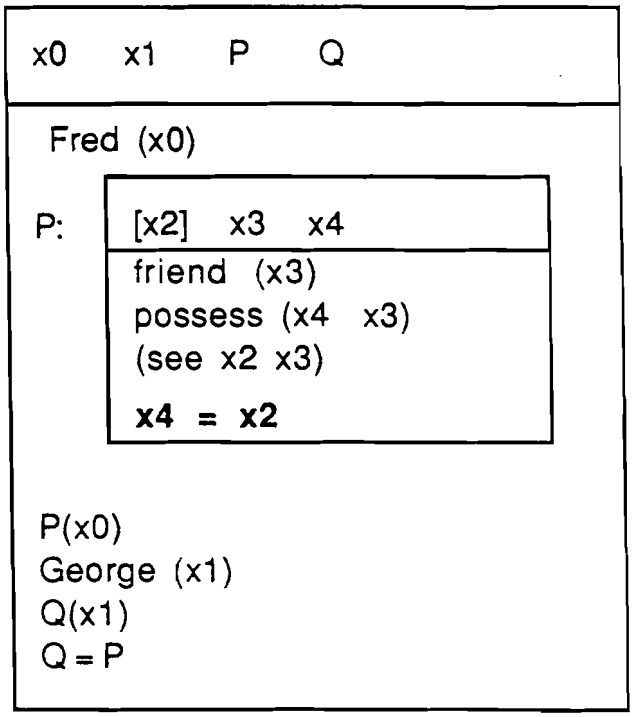

Figure 3.9: A Discourse Representation for Examples 268 and 269: *his* refers indirectly to *Fred*

phrase ellipsis. The elided sentence in 269 cannot mean *George saw a different friend of Fred's.* In contrast, the elided sentence in 268 can have this meaning. Hence, one more interpretation is available for the elided sentence in 268 than for the elided sentence in 269. One might be able to handle examples 267 and 266 by using novelty-familiarity felicity conditions to handle the differences between *a dog* and *the dog* in verb phrase ellipsis. However, the differences between examples 269 and 268 cannot be accounted for so easily. In particular, because the antecedent for *his* is *Fred*, *his friend* may not be anaphoric. Hence, there are some basic problems concerning the differences between definite and indefinite noun phrases. These differences must be accounted for if the model is to be robust. We will discuss the differences between definites and indefinites in the next chapter. There, we will demonstrate that our approach generates very different meanings for the elided sentences in 269 and 268.

One other problem with Klein's model is its failure to handle Bach-Peters sentences, like the one in 230.

Example 230

(The boy who wrote her$_j$)$_i$ kissed (the girl who loved him$_i$)$_j$.

The Discourse Representation for this sentence is shown in Figure 3.10. There is no way for the discourse referent corresponding to the pronoun *her* (i.e., *x2*) to access the discourse referent corresponding to the object of the verb phrase (i.e., *x4*), because *x2* is not contained in the box labelled *R*, which contains *x4*[25].

Because quantifier scoping information must be obtained before the representation of a sentence is given in Discourse Representation Theory, the approach violates either compactness or modularity. Certainly a Discourse Representation is not a logical form, however, for it to be useful for the incremental comprehension of language, its structure must be made less susceptible to quantifier scoping. In our approach, the structure of logical form is not affected by quantifier scoping until the final meaning is available. Despite the fact that the meaning of a restriction on a universal differs from that of an indefinite, this information is not used until later in processing (i.e., the meaning of the restriction is expanded only when we are ready to settle on a final interpretation). Additionally, in contrast to Discourse Representation Theory, our approach handles definites and indefinites quite differently, and thus we are able to handle both examples 266 and 267, as well as 268 and 269, as we will show in the next chapter.

[25]We are assuming that all sentences, even relative clauses, must have their subjects abstracted. Because of this fact, accessibility is a problem in crossing reference sentences. For that matter, accessibility in donkey sentences is impaired.

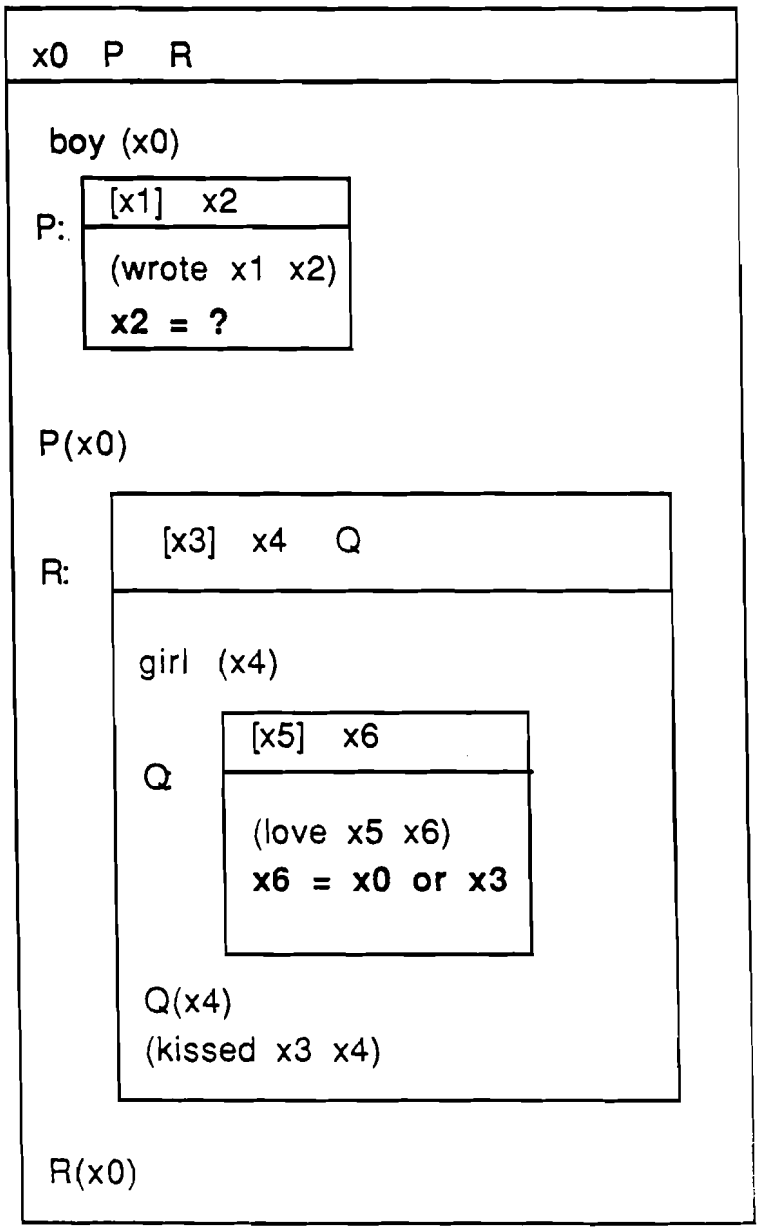

Figure 3.10: A Discourse Representation for Example 230

Chapter 4

Indefinite Noun Phrases

4.1 Introduction

In this chapter, we discuss the representation of indefinite noun phrases in logical form. We motivate this representation using our constraints on logical form proposed in Chapter one, as well as linguistic evidence on the nature of indefinite noun phrases. Then, we propose an initial representation for indefinites, as well as ways to update the initial indefinite representation so that their wide range of behaviors can be accounted for.

4.2 Indefinite Noun Phrases: Linguistic Evidence

In this section, we characterize the ways that indefinite noun phrases behave in English. To achieve this end, we compare and contrast the behavior of indefinite noun phrases with definite noun phrases. We also consider how well an existential models the behavior of an indefinite.

Indefinite noun phrases, unlike definites, are never anaphoric[1]. However, like definites, their structure can be quite complex. Pronouns and other noun phrases can be embedded in an indefinite noun phrase. The meaning of an indefinite, like the meaning of a definite, is affected by the meanings of embedded noun phrases. For example, the antecedents chosen for pronouns embedded in an indefinite affects the meaning of the indefinite. Consider example 271.

Example 271

Every man$_i$ saw a friend of his$_i$.

In this example, the embedded pronoun *his* is bound by *every man*, so *every man* distributes over the indefinite. Example 271 is not that different from example 272.

[1] Even if two indefinites denote the same thing, we do not consider one of them to be anaphoric. For example:

Example 270
Sally walked up to Bill carrying a kite.
She asked him whether he would trade his frog for a kite and a top.

Even if Bill sees the kite Sally is carrying and she asks him whether he would trade for *a kite*, the second instance of *a kite* in the story is not anaphoric. In fact, Bill might infer that the kite Sally is carrying is the kite to be traded. However, she might have some other kite in mind.

Example 272

```
Every man; saw his; friend.
```

In examples 271 and 272, once the antecedent for the pronoun is specified, the behaviors of the definite and indefinite noun phrases are similar.

Embedded quantified noun phrases similarly affect the meanings of definites and indefinites. Consider example 273.

Example 273

```
Fred saw a dog who bit every man.
```

When *every man* is embedded in a relative clause attached to an indefinite, it cannot distribute over the indefinite because of the complex noun phrase constraint. Hence, *a dog who bit every man* seems to denote a single dog. If we make the relative head *the dog* instead of *a dog* (see example 274), not much changes.

Example 274

```
Fred saw the dog who bit every man.
```

Also, when a universal is embedded in a prepositional phrase attached to a definite or indefinite noun phrase, similar ambiguities arise. Consider example 275.

Example 275

```
Fred saw a picture of every child in the class.
Meanings:
1. ∀x: (and (child x) (in x ((def2) | (class (def2))))) ∃y: (picture-of x y)
     (see Fred y)
   ; Fred saw a picture of each individual child.
2. ∃y: (∀x: (and (child x) (in x ((def2) | (class (def2))))) (picture-of x y))
     (see Fred y)
   ; Fred saw a group shot of the class.
```

There are two possible readings for the sentence in example 275. Either the universal distributes over the indefinite or it does not. A similar set of readings arises if we replace *a picture of every child in the class* by *the picture of every child in the class*, as shown in 276.

Example 276

```
Fred saw the picture of every child in the class.
Meanings:
1. ∀x: (and (child x) (in x ((def2) | (class (def2)))))
    (see Fred ((def1 x) | (and (picture (def1 x))
                               (possess x (def1 x)))))
    ; Fred saw each child's picture.
2. (see Fred ((def1) | (and (picture (def1))
                            ∀x: (and (child x) (in x ((def2) |
                                                      (class (def2)))))
                            (possess x (def1 x)))))
   ; Fred saw the group shot of the class.
```

Hence, there are many similarities between structurally complex definite and indefinite noun phrases. The meaning of each type of noun phrase is affected by the meanings of its embedded noun phrases and by whether or not embedded noun phrases distribute over it.

Despite the structural similarities between definites and indefinites, indefinites are different than definites. We illustrate these differences with several examples. First, consider how negation affects indefinites differently than definite noun phrases. For example, compare example 277 with example 278.

Example 277

```
Fred did not see a woman.
Meanings:
1. ∃x: (woman x) Not(see Fred x)
2. Not ∃x: (woman x) (see Fred x)
   or equivalently
   ∀x: (woman x) Not(see Fred x)
```

Whenever there is negation in a sentence with an indefinite, two meanings of the sentence are possible. If the negation does not have scope over the indefinite, then reading 1 seems reasonable. In contrast, if the negation has scope over the indefinite, then reading 2 is expected. In contrast, notice how the meaning of the sentence changes when we replace *a woman* in 277 by *the woman* in example 278.

Example 278

```
Fred did not see the woman.
```

Definites do not seem to be affected by negation in the same way as indefinites. Because of this, we can initially represent definites as functions in logical form. The definite *the woman* in example 278 can be represented immediately as a function. However, because the meaning of an indefinite is affected by whether negation has scope over it or not, it cannot be initially represented as a function. If we represent the indefinite in 277 as a function before deciding whether the negation has scope over it, then the second reading could not be expressed.

Next, consider a minor modification of examples 271 and 272. Assume now that the antecedent for *his* is some noun phrase outside of the sentence. Consider the indefinite case first.

Example 279

Every man$_i$ saw a friend of his$_k$.

Even though the antecedent for *his* is not *every man*, *every man* can still distribute over *a friend of his*. In contrast, consider the definite in example 280.

Example 280

Every man$_i$ saw his$_k$ friend.

Given that the antecedent for *his* is not *every man*, a distributive reading of *his friend* is impossible.

Another difference between definites and indefinites arises in verb phrase ellipsis. Consider example 266, already discussed in Section 3.4.5 of Chapter three.

Example 266

Fred saw a dog.
George did too.
Meanings:
1. George saw the same dog that Fred saw.
2. George saw a different dog than Fred saw.

When an indefinite noun phrase occurs in the verb phrase of a trigger sentence, an ambiguity arises in the meaning of the indefinite in the elided sentence. George can possibly see a different dog than Fred saw. However, if the indefinite in 266 is replaced by a definite noun phrase, then a very different situation arises, as can be seen in example 267.

Example 267

Fred saw the dog.
George did too.
Meanings:
George saw the same dog that Fred saw.

In this case, Fred and George must both see the same dog. Furthermore, these differences persist even when the indefinite and definite noun phrases contain embedded pronouns whose antecedent is the syntactic subject of the trigger sentence. Consider example 268, discussed in Section 3.4.5 of Chapter three.

Example 268

Fred$_i$ saw a friend of his$_i$.
George$_j$ did too.
Meanings:
1. George saw the same friend of Fred's.
2. George saw a different friend of Fred's.
3. George saw a friend of George's.

When *his* is directly dependent on the syntactic subject of the trigger sentence in 268, there is no guarantee that Fred and George saw the same friend. All we know is that they both saw some friend of Fred's. Compare this example with a similar definite example (i.e., 269), also discussed in Chapter three.

Example 269

Fred$_i$ saw his$_i$ friend.
George$_j$ did too.
Meanings:
1. George saw Fred's friend.
2. George saw George's friend.

In 269, given that *his* is directly dependent on the syntactic subject of the trigger sentence, Fred and George must both see the same friend. These examples of verb phrase ellipsis suggest that the representation for indefinites should differ from our representation of definites, discussed in Chapter three.

Indefinite noun phrases have traditionally been represented as existentially quantified and restricted variables. Consider the sentence in 281.

Example 281

A man walked.

∃x: (man x) (walk x)
equivalent to:
∃x (and (man x) (walk x))

In this example, the existential seems a reasonable representation for the indefinite, though it may not be necessary to capture the meaning of the indefinite. Certainly, the indefinite could be represented as a constant, if we add Heim's [18] novelty-familiarity felicity condition. There are, however, many examples which strongly suggest the necessity of an existential operator to model certain indefinite behaviors. Example 277 is such an example.

Example 277

Fred did not see a woman.
Meanings:
1. ∃x: (woman x) Not(see Fred x)
2. Not ∃x: (woman x) (see Fred x)
   or equivalently
   ∀x: (woman x) Not(see Fred x)

Whenever there is negation in a sentence with an indefinite, two meanings of the sentence are possible. If the negation does not have scope over the indefinite, then the first reading seems reasonable. In contrast, if the negation has scope over the indefinite, then the second reading is expected. The second reading provides strong evidence for initially representing an indefinite with an existential operator. Under the scope of negation, the indefinite becomes a universal. Because the meaning of an indefinite is affected by whether negation has scope over it or not, and because the existential quantifier captures this aspect of indefinite behavior, we cannot easily rule out an existential representation for indefinites.

Another interesting indefinite behavior concerns the ambiguity that arises when an indefinite and a universal occur in the same clause. Consider example 282.

Example 282

```
Some man saw every woman.
Meanings:
1. ∃x: (man x) ∀y: (woman y) (see x y)
2. ∀y: (woman y) ∃x: (man x) (see x y)
```

The sentence in example 282 has two possible meanings. Either the same man saw each woman (the existential has wide scope over the universal) or the man depends on which woman is being seen (the universal has scope over the existential). These two meanings are consistent with the representation of an indefinite as an existentially quantified variable. In contrast, consider example 283.

Example 283

```
The man saw every woman.
```

The sentence in example 283 has a single reading. The universal cannot distribute over the definite subject in this example, even if the definite had contained an embedded pronoun (the universal could not bind such a pronoun). The behavior of the definite in 283 is easily modeled by replacing the definite with a function with no arguments. In fact, we provide the initial composite representation for the subject *the man* without considering the quantifiers in the verb phrase. However, as example 282 demonstrates, the behavior of an indefinite subject can be affected by a wider range of quantified noun phrases than can directly affect the meaning of a definite subject. Hence, the final meaning for an indefinite subject cannot be provided without considering the effect of quantifiers contained in the verb phrase. The effect that a universal noun phrase has on an indefinite subject can be modeled as a quantifier scope ambiguity.

Also, consider how a quantificational representation captures the meaning of an indefinite when it is contained in the trigger verb phrase in verb phrase ellipsis. Consider example 266.

Example 266

```
Fred saw a dog.
George did too.
Meanings:
1. George saw the same dog that Fred saw.
2. George saw a different dog than Fred saw.
```

This example can be handled if we represent indefinites using existential quantifiers. Consider the representations for the meanings of the trigger and elided sentences:

Example 284

$((\text{def}_1) \mid (\text{name } (\text{def}_1) \text{ Fred}))$, $\lambda(x)[\exists y \text{ (and (dog } x)(\text{see } x\ y))]$
$((\text{def}_2) \mid (\text{name } (\text{def}_2) \text{ George}))$, $\lambda(x)[\exists y \text{ (and (dog } x)(\text{see } x\ y))]$

Because of the existential quantifier's meaning, there is no requirement that the dogs in the trigger and elided sentences are the same, though they could be. Hence, both meanings of the elided sentence in 266 are provided by using existential quantifiers to represent indefinites in a trigger verb phrase.

We can also provide reasonable meanings for pronouns whose antecedents are indefinites using an

existential representation for the indefinites. In particular, pronouns whose antecedents are indefinites in the same sentence can be replaced with the existential variables of their antecedents. For example:

Example 285

Fred asked a professor$_i$ to describe her$_i$ research.

As discussed in Chapter two, we simply equate the pronoun function representing *her* with the existential variable of *a professor*. We can also provide reasonable meanings for pronouns whose antecedents are indefinites in previous sentences without giving up the existential representation. Consider the two sentences in example 286.

Example 286

A dog$_i$ ran into the street.
A car hit it$_i$ and it$_i$ was killed.

Notice that *a dog* can be the antecedent for *it*. Though we do not allow existential quantifiers to bind across sentences, we can use Webber's [49] method to construct discourse entities (once a sentence is processed) for indefinites. For example, in example 286, we can construct a discourse entity for *a dog* which is consistent with the singular pronouns in the second sentence.

However, someone might question whether discourse entity construction is necessary for indefinites. Maybe indefinites should bind a pronoun in a subsequent sentence because the indefinite in 286 is the antecedent of a singular pronoun. However, there are several reasons for not allowing such a binding. First, universals cannot bind across sentences, as example 287 shows.

Example 287

Every dog$_i$ ran into the street.
*A car hit it$_i$ and it$_i$ was killed.

Notice that *every dog* cannot be the antecedent for the pronoun *it*. Additionally, though universals cannot bind a pronoun in another sentence, they can be the antecedent for a plural pronoun in another sentence. Consider a slight modification of example 287, shown in 288.

Example 288

Every dog$_i$ ran into the street.
A car hit them$_i$ and they$_i$ were killed.

By using Webber's [49] method for constructing discourse entities (once a sentence is processed), we can construct a discourse entity for the universal noun phrase *every dog* compatible with the plural pronouns. Hence, example 288 is felicitous because the pronouns match their antecedent's discourse entity on number. However, example 287 is not felicitous because the pronouns do not match their antecedent's discourse entity on number. Therefore, though universals cannot bind across sentences, they can provide discourse entities compatible with pronouns in other sentences. Third, we decided in Chapter two to model intersentential anaphora by replacing pronoun functions with discourse entities. Discourse entity construction is necessary to handle example 288. Hence, given that we must construct discourse entities to handle some examples, we should use that approach for homogeneity. Finally, not every singular indefinite can be the antecedent for a

singular pronoun in another sentence. For example:

Example 289

Some man saw every incident.
He was not impressed.

If *some man* is under the scope of *every incident*, then *some man* cannot be the antecedent for *he*. The decision about whether an indefinite can be the antecedent for a singular pronoun in another sentence depends on what has scope over that indefinite. Hence, we do not allow existentials to bind across sentences. Instead, following Webber, we construct discourse entities for existentials in a sentence once the meaning of the sentence is available. These discourse entities provide the meanings of pronouns whose antecedents are indefinites in a previous sentence.

We have demonstrated how the existential representation for an indefinite is useful for modeling the effects that negation and other quantified noun phrases have on the meaning of an indefinite. Additionally, the existential operator is useful for capturing the ambiguity that arises in verb phrase ellipsis when an indefinite is included in the verb phrase of the trigger sentence. These behaviors together provide support for using an existential operator to represent indefinites. However, an existential representation does not handle all of the behaviors of an indefinite. There are several examples that deserve mention.

Verb phrase ellipsis poses a problem for the existential representation of indefinites. When an indefinite subject in a trigger sentence is represented using an existential operator, an ill-formed representation for the strict reading of the elided sentence is generated. To demonstrate this fact, consider example 290.

Example 290

A man$_i$ saw his$_i$ dog.
A policeman$_j$ did too.
Meanings:
1. A policeman saw the man's dog.
2. A policeman saw his own dog.

If we represent indefinites using existential quantifiers, the trigger sentence is initially represented as shown in 291.

Example 291

A man saw his dog.

$\exists$x: (man x) x, λ(y)(see y ((def_1 y) | (and (dog (def_1 y))
(possess (his_2 y) (def_1 y)))))

Now if we decide that the antecedent for *his* is *a man*, then we update logical form as shown in 292.

Example 292

A man$_i$ saw his$_i$ dog.

```
∃x: (man x) x, λ(y)(see y ((def1 y) | (and (dog (def1 y))
                                           (possess (his2 y) (def1 y))
                                           (or (= (his2 y) y)
                                               (= (his2 y) x)))))
```

Depending on the intended meaning of the pronoun, two different trigger representations result. Each trigger representation provides a different representation of the elided sentence. However, only one of the representations of the elided sentence is well-formed given the representation of the subject. To see this, first suppose that *his* refers indirectly to *a man*. Then the trigger sentence is reduced to the logical form shown in 293.

Example 293

A man$_i$ saw his$_i$ dog.

```
∃x: (man x) x, λ(y)(see y ((def1 y) | (and (dog (def1 y))
                                           (possess (his2 y) (def1 y))
                                           (= (his2 y) y))))
```

The representation of the trigger verb phrase is then used to provide the meaning of the elided sentence shown in 294.

Example 294

A policeman did too.

```
∃z: (policeman z) z, λ(y)(see y ((def1 y) | (and (dog (def1 y))
                                                 (possess (his2 y) (def1 y))
                                                 (= (his2 y) y))))
; A policeman saw his own dog.
```

Hence, we are able to derive the sloppy reading of the elided sentence. On the other hand, suppose that the pronoun *his* refers directly to *a man*, then the trigger sentence is reduced to the logical form shown in 295.

Example 295

A man$_i$ saw his$_i$ dog.

```
∃x: (man x) x, λ(y)(see y ((def1 y) | (and (dog (def1 y))
                                           (possess (his2 y) (def1 y))
                                           (= (his2 y) x))))
```

There is certainly nothing wrong with this representation of the trigger sentence. However, if we use the representation of the verb phrase in an attempt to provide the strict meaning of the elided sentence, an ill-formed representation results, as shown in 296.

Example 296

```
A policeman did too.

∃z: (policeman z) z, λ(y)(see y ((def1 y) | (and (dog (def1 y))
                                                 (possess (his2 y) (def1 y))
                                                 (= (his2 y) x))))
; Ill-formed representation, x is unbound.
```

Notice that the variable x is not bound in this sentence. Hence, when we represent indefinite subjects as quantified variables, we can only provide one well-formed meaning for the elided sentence in 290, despite the fact that there are two meanings for that sentence.

The meaning of an elided sentence is ambiguous when the antecedent for a pronoun in the trigger verb phrase is an indefinite subject (as shown in example 290). However, when the subject is a universal (as in example 297), the ambiguity disappears.

Example 297

```
Every man_i loves his_i mother.
Every boy_j does too.
Meanings:
Every boy loves his own mother.
```

When the antecedent of the pronoun in the trigger verb phrase is a universal subject, only the sloppy reading of the elided sentence is possible. A strict reading is impossible. Hence, existential subjects have a broader range of behaviors than a universal subject. Because the indefinite subject in example 290 gives rise to a strict meaning of the elided sentence, indefinite subjects act similarly to definite subjects (as in example 103). Though one cannot say that an existential quantifier binds across sentences, there is the possibility that an indefinite, initially represented as an existentially quantified variable, could be converted into some function later in processing (and that function may then be accessible to a pronoun in another sentence).

Because of examples like 290, Webber [49] proposes that indefinite subjects should be represented as discourse entities, while indefinites in the verb phrase should be represented using an existential operator. Though her solution handles example 290, it does not handle example 298[2].

Example 298

```
A flag was hanging before each window.
A windsock was too.
Meanings:
1. The same flag was hanging before each window.
   The same windsock was hanging before each window.
2. For each window a flag was hanging before it.
   For each window a windsock was hanging before it.
```

This example suggests that indefinite subjects should not be automatically treated as constants. While indefinite behaviors cannot always be captured by an existential operator, we must not throw the existential operator away too soon. This example demonstrates that a universal operator corresponding to a noun phrase in a trigger verb phrase can have scope over an indefinite subject.

[2]Reading 2 in 298 is the preferred reading. However, with an appropriate context, Reading 1 can also be reasonable.

Additionally, the scope decision made for the trigger subject must be inherited by the subject of the elided sentence (as observed by Hirschbühler [23] and Cormack [14]).

One other indefinite behavior presents a problem for a quantificational representation of an indefinite. Consider example 265.

Example 265

Every miner who owns a donkey$_i$ beats it$_i$.

No one has difficulty understanding that the antecedent for *it* is *a donkey*. However, because of the complex noun phrase constraint, the existential operator corresponding to *a donkey* cannot bind the pronoun. Notice that when a universal is contained in a relative clause attached to another universal, as shown in 299, the embedded universal quantifier cannot bind the pronoun *it*.

Example 299

***(Every miner who owns every donkey$_j$)$_i$ beats it$_j$.**

Because the complex noun phrase constraint prevents any quantifier from raising out of a relative clause to bind a matrix pronoun, *it* cannot be bound by *every donkey*. The complex noun phrase constraint should also prevent *a donkey* from binding *it* in 265. Additionally, we cannot assume that *a donkey* denotes some particular animal. *A donkey* can be the antecedent for *it* even though the universal has scope over the indefinite. Hence, we cannot construct a discourse entity for the indefinite compatible with the singular pronoun. However, the sentence in 265 is easily understood by English speakers. Hence, it is a behavior our approach should handle.

In summary, indefinites and definites, though they share many properties, differ in many respects. Indefinites are affected by negation. Indefinites act differently in verb phrase ellipsis than definites. Finally, indefinites are more widely involved in quantifier scope ambiguity than definites. These differences preclude using the same representation for both types of noun phrases. In fact, the intrasentential behavior of an indefinite can be easily modeled by using the existential operator. However, when we consider the behavior of an indefinite across sentences or across clauses (i.e., to handle verb phrase ellipsis or donkey sentences), then the existential cannot cover all of the necessary behaviors. In the next section, we discuss how indefinites should initially be represented in logical form. Then once additional information is available, we suggest modifications of this initial representation to handle the examples that are difficult for an existential representation.

4.3 The Initial Representation of Indefinites and Its Modifications

Because of the ways that indefinites behave, particularly with respect to negation, we represent indefinites initially as existentially quantified and restricted variables. Consider the initial representation of the indefinite in the following example:

Example 300

Fred saw a dog.

((def$_1$) | (name (def$_1$) Fred)), λ(x)(saw x [∃y: (dog y) y])

Notice that following Schubert and Pelletier [46] and Allen [1], we leave the quantifier in the predicate-argument structure for the verb phrase (see Appendix B for the semantics of a quantified term). .Because the initial representation of the indefinite is provided using only syntactic information and knowledge about how to map arguments into the predicate-argument structure, our initial representation for an indefinite is compatible with the modularity constraint. Since we do not initially indicate quantifier scoping information, no contextual information is required to provide the initial representation for an indefinite. However, once quantifier scoping information is available, we must provide a way to add this information to the logical form for the sentence. As we will soon demonstrate, we specify quantifier scoping information using a mechanism similar to Allen [1].

Now that we have an initial representation, what can we do with it to model all of the behaviors that indefinites show? The existential operator is only necessary until we decide what has scope over the indefinite. Once this information is available, there is little reason to continue representing an indefinite with an existential operator. If there is some way to transform the existential into a form more compatible with other indefinite behaviors and that transformation only limits the meaning of the initial indefinite (in keeping with the formal consistency constraint from Chapter one), then such a transformation is desirable.

There is precedence for converting existentially quantified variables into functions. When a predicate calculus sentence is skolemized, existential variables are replaced by functions of all the variables corresponding to universal operators that have scope over the existential operator. In order to skolemize a sentence, we must be able to resolve quantifier scope ambiguity. Once ambiguity is eliminated, we must determine which variables are universal and which are existential. The type of quantifier switches for each negation that has scope over the quantifier. Once scoping is specified, each existentially quantified variable can be replaced by a function whose argument list consists of all of the universally quantified variables that have scope over its operator. To demonstrate how an existential is replaced by a function during skolemization, consider example 282.

Example 282

```
Some man saw every woman.
Meanings:
1. ∃x: (man x) ∀y: (woman y) (see x y)
2. ∀y: (woman y) ∃x: (man x) (see x y)
```

Each of the meanings in 282 is preserved when the existential variables are replaced by a function of all the variables corresponding to universal operators that have scope over the existential operator. The first representation in 282 is converted into an equivalent form where variable x is replaced by a function with no arguments (see the last formula in 301).

Example 301

```
∃x: (man x) ∀y: (woman y) (see x y)   ; specify scoping
equivalent to:
∃x (and (man x)                       ; expand syntactic sugar
        ∀y (if (woman y) (see x y)))  ; determine quantifier type
equivalent to:
(and (man (indef22))                  ; replace existential variables with
     ∀y (if (woman y)                 ; function
            (see (indef22) y)))
```

We describe each step of the conversion process. Once quantifier scoping is known, the representation of the sentence's meaning (the first formula in 301) is converted into a form where syntactic sugar is expanded (the second formula in 301). For our conversion process to proceed correctly, we must keep in mind that the meaning of *if* in (if p q) is logically equivalent to (or (not p) q). An existential in the scope of an odd number of negations cannot be replaced with a function. However, because the existential in the second formula in 301 is not contained in the if clause, it is not in the scope of negation. Finally, each existentially quantified variable is replaced by a function of all of the variables corresponding to universal operators that have scope over their existential operator. In this case, no universal operator has scope over the existential, and so all of the existential variables (i.e., x) are replaced by a function with no arguments (or a constant). This substitution gives the third formula in 301. Now, consider how the second representation in 282 is converted (shown in example 302).

Example 302

```
∀y: (woman y) ∃x: (man x) (see x y)    ; specify scoping
equivalent to:
∀y (if (woman y)                       ; expand syntactic sugar
       ∃x (and (man x)                 ; determine quantifier type
               (see x y)))
equivalent to:
∀y (if (woman y)                       ; replace existential variables with
       (and (man (indef23 y))           ; function
            (see x (indef23 y))))
```

In this case, the existential variable x is replaced by a function of y, since ∀y has scope over ∃x.

There are several advantages gained by replacing existential variables by functions. First, it provides us with a way to indicate quantifier scoping in a sentence containing only universals and indefinites. Additionally, we do not have to move quantifiers around. This mechanism is similar to Allen's [1] method of indicating quantifier scoping. Also, it is not possible to replace universal variables with functions. Hence, it may be precisely what is needed to model the differences between universals and indefinites in English. Furthermore, once quantifier scoping information is available, replacing existential variables by functions is a meaning preserving operation. The formal consistency constraint requires that any logical form update should only further specify logical form, not change it. Finally, a functional representation allows us to handle many of the examples that were problems for an existential representation of indefinites. In the rest of this section, we demonstrate how replacing existentials with functions allows us to handle the examples that were problems for an existential representation of indefinites. But before doing so, we examine how this functional conversion of indefinites is affected by the lambda operator in order to handle verb phrase ellipsis.

To handle verb phrase ellipsis, we must determine whether universals have scope over an indefinite and also whether any lambda operators do. If we do not consider lambda operators when converting existentials to functions, then example 266 would not be handled.

Example 266

```
Fred saw a dog.
George did too.
Meanings:
1. George saw the same dog that Fred saw.
2. George saw a different dog than Fred saw.
```

The initial representation of the trigger sentence is shown in 303.

Example 303

```
Fred saw a dog.

((def1) | (name (def1) Fred)), λ(x)(saw x [∃y: (dog y) y])
```

Notice that there are no universals in the trigger sentence. So if we skolemize the existential corresponding to *a dog* without considering the lambda operator, we would replace the existential variables with a function with no variables.

Example 304

```
Fred saw a dog.

((def1) | (name (def1) Fred)), λ(x)(saw x ((indef2) | (dog (indef2))))
equivalent to:
((def1) | (name (def1) Fred)), λ(x)(and (saw x (indef2)) (dog (indef2)))
```

The vertical bar on the indefinite function is equivalent to the colon for an existential quantifier. The restriction is expanded as shown in the last line of 304. The problem with the representation in 304 is that the verb phrase can only be used to generate the first meaning of the elided sentence (shown in 305).

Example 305

```
George did too.

((def2) | (name (def2) George)), λ(x)(saw x ((indef2) | (dog (indef2))))
equivalent to:
((def2) | (name (def2) George)), λ(x)(and (saw x (indef2)) (dog (indef2)))
; George saw the same dog as Fred saw.
```

In this meaning of the elided sentence, George must see the same dog as Fred saw.

To handle examples like 266, we must not only take into account whether universal operators have scope over an existential, but we must also decide whether lambda operators do. The representation in 303 contains no universal operators and a single lambda operator, $\lambda(x)$. Suppose we decide that $\lambda(x)$ has scope over the existential operator in 303, then the existential variables are replaced with a function of x, as shown in 306.

Example 306

```
Fred saw a dog.

((def1) | (name (def1) Fred)), λ(x)(saw x ((indef2 x) | (dog (indef2 x))))
equivalent to:
((def1) | (name (def1) Fred)), λ(x)(and (saw x (indef45 x))
                                        (dog (indef45 x))))
```

Using the verb phrase representation from 306, we are able to provide the second meaning for the elided sentence of 266, shown in 307.

Example 307

```
George did too.

((def2) | (name (def2) George)), λ(x)(saw x ((indef2 x) | (dog (indef2 x))))
equivalent to:
((def2) | (name (def2) George)), λ(x)(and (saw x (indef45 x))
                                         (dog (indef45 x))))
; George saw a potentially different dog than Fred did.
```

In contrast, if the lambda does not have scope over the existential operator, then the trigger sentence is represented as shown in 304. And as we have already shown, this representation of the trigger sentence can be used to provide the first reading of the elided sentence in 266 (shown in 305). Hence, by determining whether a lambda operator has scope over an indefinite, we can provide the two different readings for the elided sentence in 266.

When an indefinite is represented as an existentially quantified variable and is not in the scope of negation, it can be replaced by a function. However, additional information about the indefinite noun phrase must be gathered before such a transformation is performed. In fact, six types of information must be available before the conversion is possible.

1. Determine the antecedents of pronouns and anaphoric definites embedded in the restriction of the existential operator.

2. Convert embedded indefinites to functional form.

3. Determine the necessary arguments for any embedded definites. Replace each definite's initial function with a function over only the necessary arguments.

4. Determine whether quantified noun phrases that are embedded in the restriction of the existential and are not subject to the complex noun phrase constraint have scope over it.

5. Determine whether the lambda operators that can have scope over the existential (because of position) do have scope over it. Sometimes a lambda operator must have scope over the existential in order to bind a variable in the restriction of the existential.

6. Determine whether quantified noun phrases (not embedded in the restriction of the existential) have scope over the existential. Again, sometimes an operator must have scope over the existential to bind a variable in the restriction of the existential.

Once this information is available, all of the existential operator's variables are replaced by a function whose argument list contains those variables corresponding external operators that have scope over the existential (so long as it is not in the scope of negation). Notice that, in order to replace an existential variable by a function, anaphora resolution, definite argument reduction, and quantifier scoping information must be available. Hence, when an existential variable is replaced by a function, its meaning is pinpointed.

The representation of an indefinite as a function is very useful, especially for capturing the behavior of indefinite subjects in verb phrase ellipsis. Consider again example 290.

Example 290

A man$_i$ saw his$_i$ dog.
A policeman$_j$ did too.
Meanings:
1. A policeman saw the man's dog.
2. A policeman saw his own dog.

Remember that the initial representation for the trigger sentence in example 290 is shown in 291.

Example 291

$\exists$x: (man x) x, λ(y)(see y ((def$_1$ y) | (and (dog (def$_1$ y))
(possess (his$_2$ y) (def$_1$ y)))))

Now suppose that the pronoun resolution module informs us that the antecedent for *his* is *a man*. We must either replace the pronoun function by the lambda variable y or by the variable x. However, as we have already shown, replacing a pronoun in a trigger sentence with an existential variable corresponding to the subject does not help us to derive the strict reading of the elided sentence; an ill-formed meaning results. If, on the other hand, we replace the existential variable representing the subject with a function, we can avoid this difficulty[3]. Hence, we must determine whether any operators have scope over the indefinite subject. In this case, there are no universals and the lambda operator cannot have scope over the existential because the operator is placed to the right of the indefinite. Hence, we replace each x with a function with no arguments, as shown in 308.

Example 308

((indef$_3$)| (man (indef$_3$))),
λ(y)(see y ((def$_1$ y) | (and (dog (def$_1$ y))
(possess (his$_2$ y) (def$_1$ y))))))
equivalent to:
(and (man (indef$_3$))
(indef$_3$), λ(y)(see y ((def$_1$ y) | (and (dog (def$_1$ y))
(possess (his$_2$ y) (def$_1$ y))))))

Now, we can express the anaphoric dependence between *his* and *a man* without generating an ill-formed representation for the strict meaning of the elided sentence. The representation for the trigger sentence in which the pronoun refers directly to the subject is shown in example 309.

[3]We can assert that (his$_2$ y) should be replaced by x first, and then replace each of the existential's variables by a function, or we can replace the existential's variables first and then equate the pronoun function with the existential's function. The effect is the same, so long as the existential is converted to a functional form before the elided sentence's meaning is determined.

Example 309

```
((indef3)| (man (indef3))),
   λ(y)(see y ((def1 y) | (and (dog (def1 y))
                               (possess (his2 y) (def1 y))
                               (= (his2 y) (indef3))))))
```

The strict meaning for the elided sentence is derived from the representation of the verb phrase in 309.

Example 310

```
∃z: (policeman z)
  z, λ(y)(see y ((def1 y) | (and (dog (def1 y))
                                 (possess (his2 y) (def1 y)
                                 (= (his2 y) (indef3))))))
; A police man saw the man's dog.
```

Unlike the representation of the elided sentence in 296, this representation is well-formed. The function (indef_3) contains no unbound variables and has the same meaning in the trigger and elided sentences. Hence, by converting indefinites into functions, we are able to handle the fact that an elided sentence can receive a strict reading even when its trigger verb phrase contains pronouns whose antecedent is an indefinite subject.

The functional representation is also quite useful for determining whether a singular indefinite can be the antecedent for a singular pronoun in a subsequent sentence. Consider example 311.

Example 311

```
Every woman saw a dog_i.
It_i bit the tallest woman.
```

The antecedent for the pronoun *it* can be *a dog* only if the universal operator corresponding to *every woman* and the lambda operator corresponding to the subject do not have scope over the existential. Consider the initial representation of the first sentence.

Example 312

```
Every woman saw a dog.

∀x: (woman x) x, λ(y)(see y [∃z: (dog z) z])
```

Now, suppose that we decide that the universal has scope over the indefinite, then the logical form would be expanded, as shown in 313.

Example 313

```
∀x: (woman x) x, λ(y)(see y ((indef34 x) | (dog (indef34 x))))
```

The initial representation of the second sentence is shown in 314.

Example 314

```
It bit the tallest woman.

(it35), λ(w)(bite w ((def44 w) | (and (woman (def44 w))
                                      (tallest (def44 w)))))
```

Now suppose that we wished to assert that the antecedent for *it* is *a dog*. Using Webber's method of constructing discourse entities [49], we can create a discourse entity for ($indef_{34}$ x). However, the discourse entity created for ($indef_{34}$ x) denotes a set of dogs. Hence, the pronoun resolution module would not allow *a dog* to be the antecedent for *it*. On the other hand, if we decide that neither the lambda operator nor the universal has scope over *a dog*, then the logical form for the first sentence (shown in 312) is updated as shown in 315.

Example 315

```
∀x: (woman x) x, λ(y)(see y ((indef37)| (dog (indef37))))
```

Now, because *a dog* is represented as a function with no arguments, a discourse entity denoting a single dog is created for that noun phrase. Hence, we can update the representation of the second sentence to indicate the anaphoric relationship between *it* and *a dog* (shown in 316).

Example 316

```
It bit the tallest woman.

(and (it35), λ(w)(bite w ((def44 w) | (and (woman (def44 w))
                                          (tallest (def44 w)))))
     (= (it35) dog56))
```

This example demonstrates the usefulness of the functional form of a singular indefinite. Just because an indefinite occurs in a sentence does not mean that it can be the antecedent for a pronoun in another sentence. The scoping decisions we make determine the type of discourse entity created for the indefinite. Hence, given that pronoun resolution matches number features of anaphors with number features of possible antecedents (along with other features), we can determine whether or not the indefinite can be the antecedent for a singular pronoun in another sentence.

Now that we have examined how the functional representation for an indefinite is used to determine whether an indefinite can be the antecedent for a singular pronoun, we turn our attention to the donkey sentence in example 265.

Example 265

```
Every miner who owns a donkey; beats it;.
```

Consider the initial representation of this sentence shown in 317.

Example 317

Every miner who owns a donkey beats it.

```
∀x: (and (miner x)
         x, λ(y)(own y [∃z: (donkey z) z]))
 x, λ(w)(beat w (it58 w))
equivalent to:
∀x (if (and (miner x)
            x, λ(y)(own y [∃z: (donkey z) z]))
       x, λ(w)(beat w (it58 w))
```

In English, quantified noun phrases attached to a noun phrase by a relative clause cannot bind a pronoun in the matrix sentence. Hence, *it* is not represented as a function of *z*. Since *it* is not represented as a function of *z* we cannot replace (it_{58} w) by *z* in attempt to capture the meaning expressed in 265. However, if we replace the existential variables in the relative clause with a function, we may then be able to assert that the antecedent for *it* is *a donkey*.

To replace the variables corresponding to the existential operator with a function, we must decide whether the operator is in the scope of negation. Because the relative clause *who owns a donkey* is part of the restriction of a universal operator and the restriction is contained in an *if* clause, negation might have scope over *a donkey* (though it also might not)[4]. Given that the negation does not have scope over the existential operator corresponding to *a donkey*, we can replace the existential's variables with a function of those variables corresponding to operators that have scope over the existential. In particular, we must decide whether ∀x or λ(y) have scope over the existential. Notice that if we decide that λ(y) has scope over the existential, then *a donkey* cannot be the antecedent for *it*. However, if we decide that only ∀x has scope, we can assert the anaphoric relationship between *it* and *a donkey*. Consider how the logical form in 317 is modified to indicate that the universal has scope over the indefinite (shown in 318).

Example 318

Every miner who owns a donkey beats it.

```
∀x (if (and (miner x)
            x, λ(y)(own y ((indef22 x) | (donkey (indef22 x)))))
       x, λ(w)(beat w (it58 w)))
```

Since (it_{58} w) is consistent with a function of *x*, we can assert the anaphoric relationship, as shown in 319.

[4]It is possible that the negation introduced by *if* in the universal's meaning should not affect the indefinite in the relative clause. In English, the relative clause is an island not subject to *outside* scoping influences. Hence, the negation introduced by the *if* in the univresal's meaning may not affect the indefinite in the relative clause. The only operators that can affect the meaning of an indefinite inside a relative clause are those inside the relative clause, those that bind a pronoun in the restriction of the indefinite, plus the operator corresponding to the relative head.

Example 319

Every miner who owns a donkey$_i$ beats it$_i$.

```
∀x (if (and (miner x)
            x, λ(y)(own y ((indef22 x) | (donkey (indef22 x)))))
       x, λ(w)(and (beat w (it58 w))
                   (= (it58 w) (indef22 x))))
```

Now this reading closely captures our intuitions about the meaning of the donkey sentence in English[5]. Given the functional representation for an indefinite, our approach handles example 265. Additionally, our approach cannot provide the meaning of the sentence in example 299.

Example 299

***(Every miner who owns every donkey$_j$)$_i$ beats it$_j$.**

Because the universal cannot be converted to a functional form, *it* cannot have *every donkey* as its antecedent.

A few observations should be summarized concerning when existentials can be replaced by a function. Certainly, when an existential is in the scope of negation, the existential should not be replaced by a function. In such a case, that existential cannot be the antecedent for a pronoun unless its variable is included as an argument in the function representing the pronoun. Consider example 320.

Example 320

***Every miner who doesn't own a donkey$_i$ beats it$_i$.**

Notice that the negation in the relative clause seems to prevent *a donkey* from being the antecedent for *it*. Is it possible for a universal in the scope of negation to be converted first to an existential and then to a function so that a pronoun could have it as its antecedent? Alas, this seems to be impossible in English. For example, if we negate a universal which is subject to the complex noun phrase constraint, it cannot be the antecedent for a matrix pronoun, despite the fact that the meaning of the sentence contains an existential. For example, consider 321.

Example 321

*Every miner who did not see every donkey$_i$ beat it$_i$.

The noun phrase *not every donkey* cannot be the antecedent for *it*. This example demonstrates that logical representations alone are not enough to determine whether an existential can be the antecedent for a pronoun. We must also keep in mind the type of noun phrase initially responsible for the logical representation. Hence, we must be very careful when determining which noun phrases can be the antecedent for a pronoun. If a noun phrase is initially represented as a universal, then unless that universal can bind the pronoun (in the same sentence), it cannot be the antecedent for that pronoun. In contrast, so long as an indefinite remains an existential, even if it cannot bind the pronoun, it may be accessible to the pronoun once we determine its precise behavior and convert

[5]Our solution for example 265 has much in common with Webber's [49] parameterized individuals. Webber introduces a parameterized individual (which looks much like an indefinite function) as the antecedent for *it* in 265. However, she does not modify the initial representation of the indefinite (represented as an existential or as a constant).

it into a function.

Before we conclude this section, there is one additional example that we should discuss (i.e., example 298).

Example 298

```
A flag was hanging before each window.
A windsock was too.
Meanings:
1. The same flag was hanging before each window.
   The same windsock was hanging before each window.
2. For each window a flag was hanging before it.
   For each window a windsock was hanging before it.
```

Because the scoping decision made for the trigger sentence is mirrored in the meaning of the elided sentence, we must provide some way for the elided subject to be scoped in the same way as trigger's subject. Additionally, we must be able to express the fact that a universal in the verb phrase has scope over the syntactic subject of the trigger sentence without leaving an unbound variable in the verb phrase (resulting in an ill-formed meaning for the elided sentence). To achieve this goal, we borrow a trick from Allen [1]. Allen, rather than moving the universal quantifier with scope over the subject out of the verb phrase, uses an alternative method for marking scope. He indicates scope information by subscripting the quantifiers that an operator has scope over with the variable corresponding to that operator. We indicate scoping by converting existential variables to functions of all of the variables corresponding to quantifiers that have scope over their existential operator. Hence, we have a mechanism similar to Allen's for indicating that a quantifier in the verb phrase has scope over the subject. We must stress, however, that when we create functions with argument lists, we do not create functions over unbound variables (see Appendix B). We are very careful to avoid this for all types of functions[6]. With this in mind, consider how we handle example 298. The initial representation of the trigger sentence is shown in 322.

Example 322

```
A flag was hanging before each window.

∃x: (flag x) x, λ(y)(hang y (before [∀z: (window z) z]))
```

Now, we can convert the syntactic subject into a function in two different ways depending on whether the universal has scope over it or not. These two representations are shown in 323.

Example 323

```
A flag was hanging before each window.

1. (and (flag (indef40))
        (indef40), λ(y)(hang y (before [∀z: (window z) z]))
   ; A certain flag is hanging in front of all of the windows.
2. (and (flag (indef39 z))
        (indef39 z), λ(y)(hang y (before [∀z: (window z) z]))
   ; For every window, a flag is hanging in front of it.
```

[6] Assume that there is a routine to present the final meanings of sentences in predicate calculus. This routine would ensure that a quantifier is moved to the left of all functions that include its variable in their argument list.

Notice that in 323, the first reading indicates that the universal does not have scope over the existential. Whereas, the second reading results because the universal has scope over the existential operator. Notice we have not moved the universal, it is still inside the verb phrase representation.

In the next step, we provide the meaning of the elided verb phrase. Using the standard procedure, two meanings are generated (shown in 324).

Example 324

```
A windsock was too.

1. ∃w: (windsock w) w, λ(y)(hang y (before [∀z: (window z) z]))
2. ∃w: (windsock w) w, λ(y)(hang y (before [∀z: (window z) z]))
```

Notice that these two representations are precisely the same. However, because the scoping decision made for the subject in a trigger representation must be inherited by the subject of the derived elided sentence's meaning, the two readings of the elided sentence in 324 will differ. Once we add the information that ∀z has scope over the subject in the second meaning of the elided sentence in 324, the existential subject is replaced by a function of z, as shown in reading two in 325. On the other hand, once we add the information that ∀z does not have scope over the subject in the first meaning of the elided sentence in 324, the existential subject is replaced by a function with no arguments, as shown in the first reading in 325.

Example 325

```
A windsock was too.

1. (and (windsock (indef42))
        (indef42), λ(y)(hang-before y [∀z: (window z) z])
   ; A certain windsock is hanging in front of all of the windows.
2. (and (windsock (indef41 z))
        (indef41 z), λ(y)(hang-before y [∀z: (window z) z])
   ; For every window, a windsock is hanging in front of it.
```

The only difference between these two readings is the representation of the subject. In reading one, the subject is represented as a skolem constant. In reading two, it is represented as a function of the universal variable z. Hence, by using the parallel scope constraint, we have captured the two expected readings of the elided sentence in example 298. Each representation of the elided sentence in 325 corresponds to a trigger representation in 323.

We should point out that example 298 is not handled by Sag's model [44]. He is unable to obtain the second meaning in 298 because his alphabetic variant constraint sanctions only the first meaning. Our model handles the example because of the way we indicate quantifier scope information and the parallel scope constraint (i.e., if a quantifier in the verb phrase of the trigger sentence has scope over the trigger's subject then when we derive the meaning of the elided sentence, the same scoping must be inherited by the subject of the elided sentence). However, Sag did not create alphabetic variance for no reason. Recall example 10.

Example 10

```
Someone kissed everyone, and Sarah did too.
```

This example provides support for alphabetic variance as a method for determining possible meanings of elided sentences. Unlike example 298, the elided sentence limits the meaning of the trigger sentence. Because the elided sentence must mean *Sarah saw everyone*, the trigger sentence must mean *one particular person saw everyone.* At first glance, this example seems difficult to explain given the way we handle example 298. However, if we indicate that the parallel scope constraint requires a universal quantifier to distribute over the elided subject in the same way it distributes over the trigger subject, we can easily explain why 10 is not ambiguous. Despite the fact that we provide two possible representations of the trigger sentence in 10 (shown in 326), only one of them sanctions a meaning of the elided sentence because of the parallel distribution constraint.

Example 326

`Someone loves everyone.`

1. `(and (person (indef`$_{51}$`)) (indef`$_{51}$`), λ(y)(love y [∀z: (person z) z])`
2. `(and (person (indef`$_{50}$ `z)) (indef`$_{50}$ `z), λ(y)(love y [∀z: (person z) z])`

Now suppose that we use these two representations of the trigger sentence to provide readings for the elided sentence. These representations are shown in 327.

Example 327

`Sarah did too.`

1. `((def`$_3$`) | (name (def`$_3$`) Sarah)), λ(y)(love y [∀z: (person z) z])`
2. `((def`$_3$`) | (name (def`$_3$`) Sarah)), λ(y)(love y [∀z: (person z) z])`

Because the noun phrase *Sarah* is initially represented as a function of no variables, no quantifiers in the verb phrase can affect its meaning. The universal operator in the verb phrase cannot affect the representation of *Sarah.* That universal cannot distribute over a function with no arguments. Hence, given the parallel distribution constraint, only the first representation of the trigger sentence in 326 can be used to generate a parallel representation for the elided sentence (i.e., the first representation in 327). Notice that the subject in first representation of 326 and the subject in the first representation of 327 are not affected by the universal in the verb phrase [7].

In summary, we initially represent indefinites as existentially quantified variables in logical form. Many intrasentential behaviors of indefinites are modeled by using existential operators. However, once we consider the behavior of an indefinite across sentences or clauses (i.e., to handle verb phrase ellipsis or donkey sentences), then the existential does not cover all of the necessary behaviors. Hence, as soon as we obtain information about what has scope over the existential, we replace all existential variables with a function of all of the variables corresponding to operators that have scope over the indefinite. This functional representation is useful for explaining the variety of indefinite behaviors.

[7] The following example is unambiguous for many informants:

Example 328
A boy loves each teacher.
A girl does too.
However, some informants are aware of the ambiguity.

4.4 Discourse Representation Theory and Our Approach to Indefinites

In Chapter three, we discussed how Discourse Representation Theory (i.e., Klein's version in particular) does not explain the differences between examples 266 and 267 or examples 268 and 269. In this section, we demonstrate first how our approach handles these examples and then discuss how Klein's Discourse Representation framework can be modified to handle them as well.

In our approach, definites and indefinites are treated quite differently, not simply in how they are initially represented, but also in how they are treated to determine final meanings. Notice, that we have no counterpart of the argument reduction constraint for indefinites. Indefinites, because of scoping, can be expressed as functions of variables even when there are no unbound variables in their restrictions. Because we treat definites differently from indefinites, it is easy for us to explain the differences between 266 and 267. In section 4.3, we discussed how our approach provides two readings for the elided sentence in 266.

Example 266

```
Fred saw a dog.
George did too.
Meanings:
1. George saw the same dog that Fred saw.
2. George saw a different dog than Fred saw.
```

Initially the indefinite in the trigger sentence is represented as an existentially quantified and restricted variable (see example 303). Once we decide whether or not the lambda operator has scope over the indefinite, we convert the indefinite into its final representation (either as in example 304 or as in example 306). The lambda variable can be included in the argument list of the indefinite function even though there is no unbound lambda variable in its restriction. Depending on the decision we make, we provide different meanings for the elided sentence (compare the readings in examples 305 and 307). This is in contrast with how we would handle 267.

Example 267

```
Fred saw the dog.
George did too.
Meanings:
George saw the same dog that Fred saw.
```

Initially, the trigger sentence is represented as shown in 329.

Example 329

```
Fred saw the dog.
```

$((\text{def}_{87})\ |\ (\text{name}\ (\text{def}_{87})\ \text{Fred}))$, $\lambda(x)(\text{see}\ x\ ((\text{def}_{88}\ x)\ |\ (\text{dog}\ (\text{def}_{88}\ x))))$

If *the dog* is not anaphoric, then we must apply the argument reduction constraint to eliminate the lambda variable x from the final meaning of the definite function. After argument reduction, the trigger sentence is represented as shown in 330.

Example 330

Fred saw the dog.

((def$_{87}$) | (name (def$_{87}$) Fred)),
λ(x)(and (see x ((def$_{88}$ x) | (dog (def$_{88}$ x))))
(= (def$_{88}$ x) (def$_{89}$))

The argument reduction constraint ensures that when we determine the meaning of the elided sentence, only the correct meaning of the definite is possible[8]. The meaning of the elided sentence is shown in 331.

Example 331

George did too.

((def$_{90}$) | (name (def$_{90}$) George)),
λ(x)(and (see x ((def$_{88}$ x) | (dog (def$_{88}$ x))))
(= (def$_{88}$ x) (def$_{89}$))

Now, consider examples 268 and 269.

Example 268

Fred$_i$ saw a friend of his$_i$.
George$_j$ did too.
Meanings:
1. George saw the same friend of Fred's.
2. George saw a different friend of Fred's.
3. George saw a friend of George's.

There are three meanings for the elided sentence in 268. In contrast, there are only two meanings for the elided sentence in 269.

Example 269

Fred$_i$ saw his$_i$ friend.
George$_j$ did too.
Meanings:
1. George saw Fred's friend.
2. George saw George's friend.

Again we can easily handle the differences between these two examples because of the way we handle definites and indefinites. Definites when they are not anaphoric are expressed as functions over those variables that are unbound in their restrictions. In contrast, indefinites, which are initially represented as existentials, can be in the scope of operators that do not bind variables in their restrictions.

Because of the differences between definites and indefinites, our approach is able to provide three

[8] Nearly the same result would be obtained if we decided that *the dog* was anaphoric. Since the antecedent cannot be *Fred*, the antecedent must occur in a previous sentence (or possibly in the environment of the speaker/hearer). In this case, the antecedent would have a discourse entity associated with it (or would be represented as a function with no arguments).

meanings for the elided sentence in 268 and only two meanings for the elided sentence in 269. We demonstrate how the three readings of the elided sentence in 268 are provided using our approach. The initial representation for the trigger sentence is shown in 332.

Example 332

```
Fred saw a friend of his.

((def97) | (name (def97) Fred)),
   λ(x)(see x [∃y: (and (friend y)
                        (possess (his98 x) y)) y])
```

Given that the antecedent for *his* is *Fred*, this logical form is updated as shown in 333.

Example 333

```
Fred_i saw a friend of his_i.

((def97) | (name (def97) Fred)),
   λ(x)(see x [∃y: (and (friend y)
                        (possess (his98 x) y)
                        (or (= (his98 x) x)
                            (= (his98 x) (def97)))) y])
```

Now, before converting the existential into its functional form, we must decide which meaning of the pronoun is intended. If the first is intended, then the lambda operator must have scope over the indefinite. However, if the second is chosen, then the lambda could have scope over the indefinite, though not necessarily.

Suppose that the pronoun *his* refers indirectly to the subject. Then, the logical form is restricted as shown in 334.

Example 334

```
Fred_i saw a friend of his_i.

((def97) | (name (def97) Fred)),
   λ(x)(see x [∃y: (and (friend y)
                        (possess (his98 x) y)
                        (= (his98 x) x)) y])
```

Since $\lambda(x)$ must have scope over the existential to bind the variable x in its restriction, the existential must be a function of that variable.

Example 335

```
Fred_i saw a friend of his_i.

((def97) | (name (def97) Fred)),
   λ(x)(see x ((indef99 x) |
                (and (friend (indef99 x))
                     (possess (his98 x) (indef99 x))
                     (= (his98 x) x))))
```

This representation can be used to derive the third meaning of the elided sentence in 268, shown in 336.

Example 336

George did too.

```
((def100) | (name (def100) George)),
   λ(x)(see x ((indef99 x) |
                (and (friend (indef99 x))
                     (possess (his98 x) (indef99 x))
                     (= (his98 x) x))))
; George saw a friend of George's.
```

In contrast, assume that the pronoun *his* refers directly to *Fred.* This choice is reflected in the logical form shown in 337.

Example 337

Fred$_i$ saw a friend of his$_i$.

```
((def97) | (name (def97) Fred)),
   λ(x)(see x [∃y: (and (friend y)
                        (possess (his98 x) y)
                        (= (his98 x) (def97))) y])
```

Now, we must decide whether $\lambda(x)$ has scope over the existential. Suppose that it does. This choice is indicated by converting the existential to a function of the variable x (shown in 338)

Example 338

Fred$_i$ saw a friend of his$_i$.

```
((def97) | (name (def97) Fred)),
   λ(x)(see x ((indef99 x) |
                (and (friend (indef99 x))
                     (possess (his98 x) (indef99 x))
                     (= (his98 x) (def97)))))
```

The representation of the trigger verb phrase in 338 is used to provide the second reading of the elided sentence in 268 (shown in 339).

Example 339

```
George did too.

((def100) | (name (def100) George)),
   λ(x)(see x ((indef99 x) |
                (and (friend (indef99 x))
                     (possess (his98 x) (indef99 x))
                     (= (his98 x) (def97)))))
; George saw a different friend of Fred's.
```

On the other hand, suppose that λ(x) does not have scope over the existential in 337. This decision is indicated in the logical form of the trigger sentence in the following way:

Example 340

```
Fred_i saw a friend of his_i.

((def97) | (name (def97) Fred)),
   λ(x)(see x ((indef102) |
                (and (friend (indef102))
                     (possess (his98 x) (indef102))
                     (= (his98 x) (def97)))))
```

The representation of the verb phrase in 340 is used to provide the first reading of the elided sentence in 268 (shown in 341).

Example 341

```
George did too.

((def100) | (name (def100) George)),
   λ(x)(see x ((indef102) |
                (and (friend (indef102))
                     (possess (his98 x) (indef102))
                     (= (his98 x) (def97)))))
; George saw the same friend of Fred's.
```

Hence, we are able to provide three readings for the elided sentence in 268. In contrast, our approach provides only two readings for the elided sentence in 269. We have already discussed examples like this in Chapter three, and so we will not discuss the example further.

In Klein's [32] approach to verb phrase ellipsis, definite and indefinite representations are provided only after quantifier scoping information is available. Hence, like us, Klein provides the same kind of final representation for both definite and indefinite noun phrases (ours is a function, his is a discourse referent). One can easily see the correspondence between Klein's discourse referents in boxes and functions with argument lists. Whenever a discourse referent is created in a box introduced by a universal, it is like a function of the variable corresponding to that universal. However, Klein does not notice that he is ignoring a very important scoping issue (i.e., does the lambda have scope over noun phrases in the verb phrase or not). In particular, whenever a discourse referent is created in a verb phrase box, it is like a function of the variable corresponding to the lambda operator used to

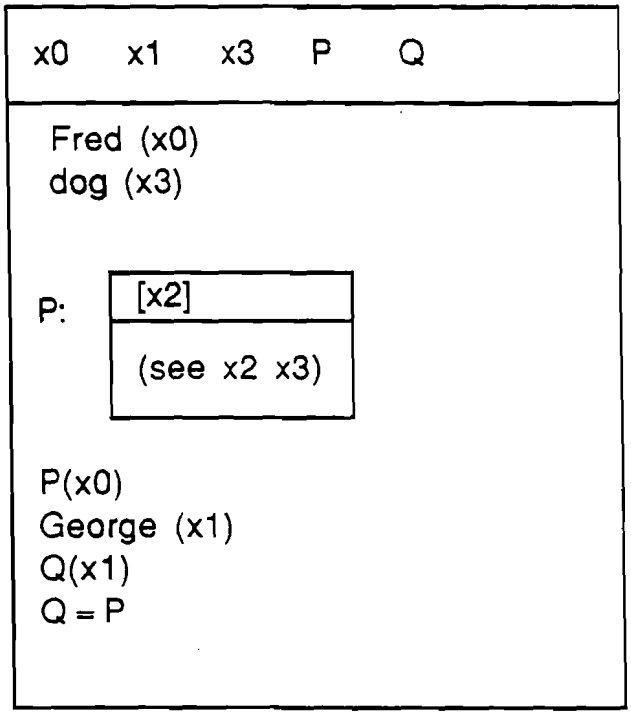

Figure 4.1: Discourse Representation for Example 267

abstract the subject. Hence, before representing a definite or indefinite in discourse representation theory, we must be able to decide whether a discourse referent should be defined inside or outside of a verb phrase box. To handle examples 266, 267, 268, and 269 in the framework of Discourse Representation Theory, we can stipulate that a discourse referent is created inside a box iff the operator responsible for introducing that box has scope over the noun phrase. We will demonstrate how adding this information helps Klein's model to handle the differences between examples 266 and 267 and examples 268 and 269.

First consider examples 266 and 267. Before providing a discourse representation for the trigger sentence in 267, we must determine all of the operators that have scope over *the dog*. To do so, antecedents for embedded anaphoric noun phrases must be determined and scoping decisions for embedded quantified noun phrases not subject to the complex noun phrase constraint must be made. In this case, there are no embedded noun phrases, and so we would limit the definite's function (assuming it is non-anaphoric) to be a function with no arguments (after the argument reduction constraint). Similarly, because nothing can have scope over the definite, it should be represented as a discourse referent created outside of the verb phrase box, as shown in Figure 4.1. Because the definite's discourse referent is created outside of the scope of the lambda box, the correct meaning of the elided sentence is provided (i.e., George saw the same dog as Fred saw). In contrast, to handle 266, we must also determine whether the lambda operator has scope over *a dog*. If it does not, then we represent it the indefinite as a function with no arguments. Similarly, in Discourse Representation Theory, the discourse referent for *a dog* should be created outside of the

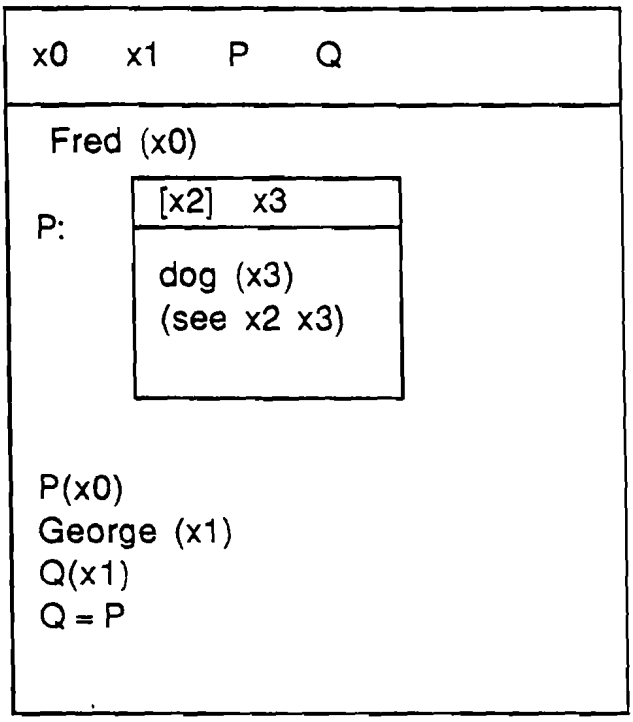

Figure 4.2: Discourse Representation for *a dog* in Example 266: the lambda operator has scope over the indefinite

verb phrase box, as shown in Figure 4.1. Hence, the strict indefinite reading of the elided sentence in 266 is captured by the discourse representation in Figure 4.1. However, if the lambda operator does have scope over the indefinite, then we represent *a dog* as a function of the lambda variable. Similarly, the discourse referent for *a dog* should be created inside of the verb phrase box. Hence, the other meaning for the elided sentence in 266 is captured by the discourse representation shown in Figure 4.2.

Now consider how these observations are used to handle examples 268 and 269. To provide a discourse representation for the trigger sentence of 269, the antecedent for *his* must be found. Assuming that *Fred* is the antecedent, we must also decide whether the pronoun refers to it directly or indirectly. Assume that *his* refers directly to the subject. In this case, the definite noun phrase *his friend* does not contain any unbound variables, and so we represent it as a function with no arguments (following argument reduction) Similarly, in Discourse Representation Theory, the definite's discourse referent should be created outside of the verb phrase box, as shown in Figure 4.3. Because the discourse referent for *his friend* is created outside of the verb phrase box (i.e., P) in Figure 4.3, the strict meaning of the elided sentence (i.e., George saw Fred's friend) is derived when Q is replaced by P. In contrast, if *his* is indirectly dependent on *Fred*, then we represent *his friend* as a function of the lambda variable. In Discourse Representation Theory, the discourse referent for *his friend* must be created inside of the verb phrase box. This representation of the verb phrase is shown in Figure 4.4. Notice that the representation of the trigger verb phrase in

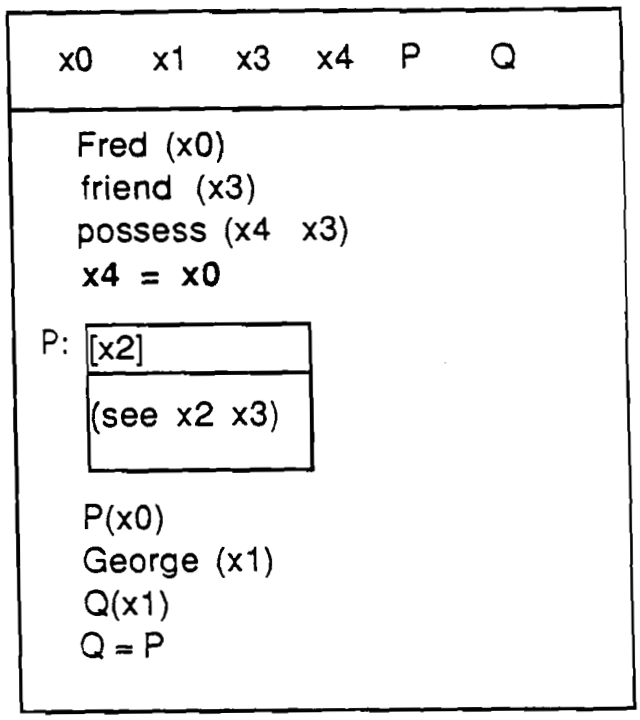

Figure 4.3: Discourse Representation for the strict meaning of the elided sentence from Example 269

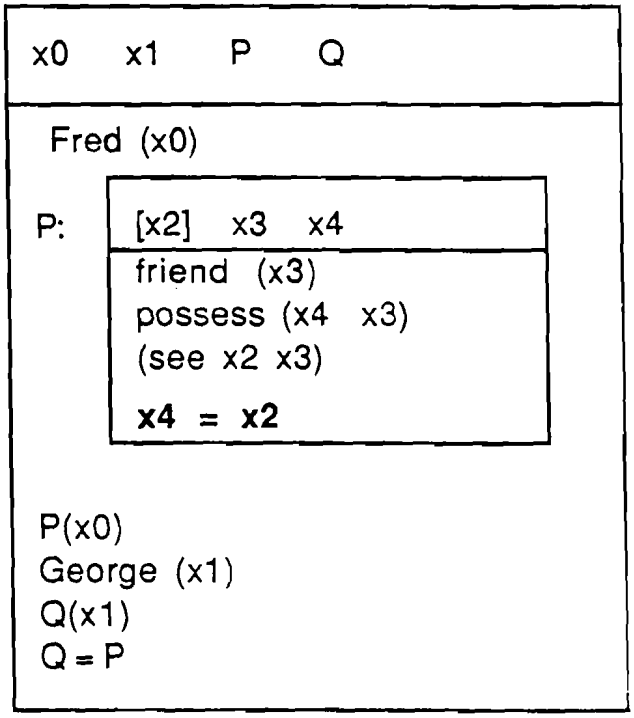

Figure 4.4: Discourse Representation for the sloppy meaning of the elided sentence from Example 269

4.4 (i.e., P) is used to provide the sloppy reading (i.e., George saw George's friend) of the elided sentence in example 269.

In contrast, consider how example 268 would be handled. As in example 269, the representation of *a friend of his* depends on decisions about how embedded noun phrases behave. Hence, the antecedent for *his* must be determined. Assuming that *Fred* is the antecedent, we must also decide whether the pronoun refers to it directly or indirectly. If *his* refers to *Fred* indirectly, we must represent *a friend of his* as a function of the lambda variable. In Discourse Representation Theory, the discourse referent for *a friend of his* should be created inside of the verb phrase box, as shown in Figure 4.4. The elided sentence would then receive the meaning indicated in that figure by equating Q with P (i.e., George saw a friend of George's). On the other hand, assuming that *his* refers to *Fred* directly, an additional decision must be made for the indefinite. Does the lambda operator of the verb phrase have scope over it? Assuming that the lambda operator does not have scope over the indefinite, we represent *a friend of his* as a function with no arguments. In Discourse Representation Theory, the discourse referent for the indefinite should be created outside of the verb phrase box, as shown in Figure 4.3. The elided sentence would then receive the meaning that George saw the same friend of Fred's as Fred saw. An additional meaning of the elided sentence is provided by assuming that the lambda operator of the verb phrase has scope over the indefinite. In this case, we represent the indefinite as a function of the lambda variable. In Discourse Representation Theory, the discourse referent for the indefinite should be created inside of the verb phrase box, as shown in figure 4.5. In this case, the meaning of the elided sentence is George saw a potentially different friend of Fred's than Fred saw.

Hence, we have demonstrated how our approach can be incorporated into Klein's Discourse Representation Theory to cover examples that were originally a problem for that approach. However, in our approach, we generate an initial composite representation for the sentence and then limit that meaning to a final meaning. In Discourse Representation Theory, at least as it is currently defined, noun phrases cannot be entered into the discourse model until scoping decisions (as well as anaphora decisions) are made.

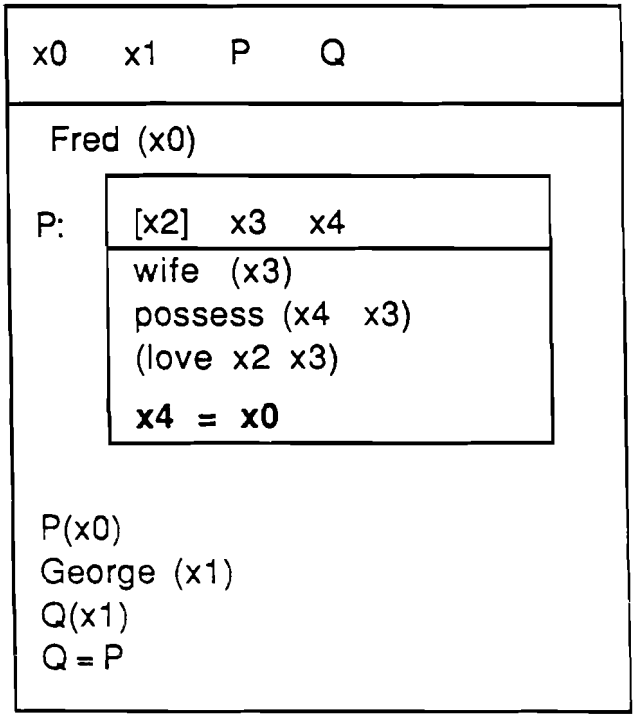

Figure 4.5: Discourse Representation for *a friend of his* in Example 268: the lambda operator has scope over the indefinite

Chapter 5

The Multiple Pronoun Constraint

In the previous chapters, we discussed only very simple examples of the sloppy identity ambiguity (i.e., examples where the antecedent for a single pronoun in a trigger sentence is the subject of that trigger sentence). Example 6 is such an example:

Example 6

Trigger Sentence: Fred$_i$ loves his$_i$ wife.
Elided Sentence: George$_j$ does too.
Meanings:
1. George loves Fred's wife.
2. George loves George's wife.

When the subject of the trigger sentence is the antecedent for a single pronoun in the same sentence, two meanings of the elided sentence result. In 6, two readings result when the antecedent for *his* is *Fred*, because the pronoun can refer to that noun phrase either directly or indirectly. Does this mean that when the antecedent for two pronouns is the subject of a trigger sentence, then four meanings of an elided sentence result? Certainly, in our approach, we generate four possible readings. However, at least in some situations, not all of the four meanings for the elided sentence are reasonable. In this section, we attempt to find a constraint to eliminate bad readings of an elided sentence when its trigger verb phrase contains two or more pronouns anaphorically dependent on the subject.

Sag [44] discusses an example of verb phrase ellipsis (i.e., example 73) in which the trigger sentence contains two pronouns whose antecedent is the subject of the sentence (the example was first noticed by Dahl [15]).

Example 73

Bill$_i$ believed he$_i$ loved his$_i$ wife.
Harry$_j$ did too.
Meanings:
1. Harry believed that Harry loved Harry's wife.
2. Harry believed that Bill loved Bill's wife.
3. Harry believed that Harry loved Bill's wife. (marginal)
4. *Harry believed that Bill loved Harry's wife.

Despite the fact that Sag's model predicts the elided sentence should have four possible meanings, it has only three reasonable meanings. In Sag's approach, the pronouns in the trigger sentence of

73 can be represented either as bound variables or as indexed pronouns. Thus, there are four ways to represent one meaning of the trigger sentence (given the antecedent for each pronoun is *Bill*). Because each representation of the trigger sentence sanctions one meaning of the elided sentence in Sag's approach, four meanings of the elided sentence are sanctioned. Each meaning of the elided sentence is listed below each representation of the trigger sentence that sanctions it in example 342.

Example 342

a. Bill$_i$, λ(x)(believed x [x, λ(y)(loved y x's wife)])
   Harry$_j$, λ(z)(believed z [z, λ(w)(loved w z's wife)])
b. Bill$_i$, λ(x)(believed x [he$_i$, λ(y)(loved y his$_i$ wife)])
   Harry$_j$, λ(z)(believed z [he$_i$, λ(w)(loved w his$_i$ wife)])
c. Bill$_i$, λ(x)(believed x [x, λ(y)(loved y his$_i$ wife)])
   Harry$_j$, λ(z)(believed z [z, λ(w)(loved w his$_i$ wife)])
d. Bill$_i$, λ(x)(believed x [he$_i$, λ(y)(loved y x's wife)])
   Harry$_j$, λ(z)(believed z [he$_i$, λ(w)(loved w z's wife)])

Despite the fact that the elided sentence meaning given by the representation in 342d is glaringly bad[1], there is no reason to rule it out based on Sag's theory. Our approach suffers from exactly the same problem.

Sag claimed that the bad reading of the elided sentence could be ruled out based on Chomsky's account [9] of Postal's crossover violations [39] combined with the fact that a pronoun cannot have a quantified noun phrase as its antecedent unless that quantified expression precedes and commands the pronoun. The constraint bars expressions where a bound variable is to the right of a pronoun indexed with the same syntactic subject (that is nothing of the form (Pro$_i$ ... x$_i$) should be allowed[2]). This constraint rules out the fourth representation of the trigger sentence (in 342d) and hence the fourth reading of the elided sentence.

If Sag's constraint covered all examples of multiple pronoun verb phrase ellipsis, we would certainly adapt it to our approach. However, Sag's constraint, despite its excellent motivations, is not correct. As evidence for this, consider example 345.

---

[1] One might think that four readings should be allowed based on the following context: *Fred loves everyone's wife. Fred believes that he loves his wife. George does too.* However, this type of example is rare, and is possibly related to the following example:

Example 343
Who loves Fred?
Well, Fred loves him.

We believe that in order to get the fourth reading of 73, the antecedent for *he* must be the *Fred* in the context sentence (i.e., *Fred loves everyone's wife*) while the antecedent for *his* is the *Fred* in the trigger sentence. As evidence for this view, it is very difficult to sanction the odd reading by manipulating the context following the elided sentence. However, such manipulation of the post-context can sway preferences for any of the other readings.

[2] Two examples that suggest this are shown in 344.

Example 344
1. *Who$_i$ did his$_i$ mother hate?
(Who x$_i$)(his$_i$ mother hate x$_i$)
2. *His$_i$ mother hated everyone$_i$.
(∀x$_i$)(his$_i$ mother hate x$_i$)

Neither of these sentences can receive their designated meaning and both have logical forms where pro$_i$ comes between either a quantifier and its trace (left by quantifier raising) or a a wh-word and its trace (left by a wh-movement).

Example 345

Bill$_i$ believed that his$_i$ wife loved him$_i$.
Harry$_j$ did too.
Meanings:
1. Harry believed that Harry's wife loved Harry.
2. Harry believed that Bill's wife loved Bill.
3. *Harry believed that Harry's wife loved Bill.
4. Harry believed that Bill's wife loved Harry. (marginal)

In Sag's approach, the trigger sentence in 345 can be represented in four different ways, and each representation of that trigger sentence sanctions one reading of the elided sentence. Each trigger representation is shown with the representation of the meaning of the elided sentence it sanctions in example 346.

Example 346

a. Bill$_i$, λ(x)(believed x [x's wife, λ(y)(loved y x)])
   Harry$_j$, λ(z)(believed z [z's wife, λ(w)(loved w z)])
b. Bill$_i$, λ(x)(believed x [his$_i$ wife, λ(y)(loved y he$_i$)])
   Harry$_j$, λ(z)(believed z [his$_i$ wife, λ(w)(loved w he$_i$)])
c. Bill$_i$, λ(x)(believed x [x's wife, λ(y)(loved y he$_i$)])
   Harry$_j$, λ(z)(believed z [z's wife, λ(w)(loved w he$_i$)])
d. Bill$_i$, λ(x)(believed x [his$_i$ wife, λ(y)(loved y x)])
   Harry$_j$, λ(z)(believed z [his$_i$ wife, λ(w)(loved w z)])

Notice that the representation of the trigger sentence in 346d (and the associated reading of the elided sentence) is ruled out based on Sag's (Pro$_i$...x$_i$) constraint, though this representation sanctions the marginally correct reading of the elided sentence (i.e., the fourth meaning in 345). Additionally, Sag's constraint predicts that the representation in 346c should sanction a reasonable reading of the elided sentence, though the reading it sanctions is bad (i.e., the third meaning in 345).

An additional failure of Sag's constraint is easily devised by passivizing the embedded clause of Sag's original example (i.e., example 73). This example is shown in 347.

Example 347

Bill$_i$ believed that his$_i$ wife was loved by him$_i$.
Harry$_j$ did too.
Meanings:
1. Harry believed that Harry's wife was loved by Harry.
2. Harry believed that Bill's wife was loved by Bill.
3. Harry believed that Bill's wife was loved by Harry. (marginal)
4. *Harry believed that Harry's wife was loved by Bill.

Given the transformation, the bad reading is semantically similar to the bad reading in 73. However, Sag's constraint does not rule out the bad reading of the elided sentence in 347. Instead, Sag's constraint rules out the representation of the trigger sentence that sanctions the marginal reading of the elided sentence. In fact, Sag was aware that there was some problem with his constraint. He pointed out in a footnote that the constraint did not quite work, giving example 348 as evidence for the failure.

Example 348

Edith$_i$ said that finding her$_i$ husband nude had upset her$_i$.
Martha$_j$ did too.
Meanings:
1. Martha said that finding Edith's husband nude had upset Edith.
2. Martha said that finding Martha's husband nude had upset Martha.
3. Martha said that finding Edith's husband nude had upset Martha.
4. *Martha said that finding Martha's husband nude had upset Edith.

The third reading of the elided sentence is not allowed based on his constraint, and yet it is an acceptable reading. The fourth reading of the elided sentence is allowed based on his constraint, though it is not an acceptable reading of the elided sentence.

Because Sag's constraint fails to handle examples 345, 347, and 348, we develop an alternative constraint. By examining a variety of examples, we attempt to determine the factors that affect multiple pronoun verb phrase ellipsis. Let us begin by examining an example of a trigger sentence containing two reflexive pronouns (shown in 349).

Example 349

Fred$_i$ told himself$_i$ about himself$_i$.
George$_j$ did too.
Meanings:
1. George told Fred about Fred.
2. George told George about George.
3. *George told Fred about George.
4. *George told George about Fred.

In this example, there are only two acceptable readings of the elided sentence. However, Sag's multiple pronoun constraint sanctions three meanings of the elided sentence in 349, namely the first, second, and fourth readings. Notice that both pronouns occur in the same clause, neither pronoun is embedded in another noun phrase, and both are anaphorically dependent on the subject *Fred*. Because the meanings of the elided sentence depend on the ways that we represent the pronouns in the trigger sentence, this example suggests that when two pronouns are anaphorically dependent on the same subject of a trigger sentence, occur in the same clause, and are not embedded in a noun phrase then they must refer to their antecedent in the same way.

Now consider an example of a trigger sentence where both pronouns occur in the same clause, are embedded in a noun phrase, and are anaphorically dependent on the trigger subject.

Example 350

Fred$_i$ showed his$_i$ mother his$_i$ dog.
George$_j$ did too.
Meanings:
1. George showed Fred's mother Fred's dog.
2. George showed George's mother George's dog.
3. George showed Fred's mother George's dog.
4. George showed George's mother Fred's dog.

In this case, all four readings seem acceptable, though based on Sag's constraint the third one

should be ruled out. The easiest way to see that both of the mixed readings are reasonable is to somehow distinguish *his mother* or *his dog* in the context following the elided sentence. For instance, the third meaning would be favored by following the elided sentence with something like, *She is always interested in seeing anyone's dog.* This example indicates that if two pronouns occur in a trigger sentence and each pronoun is embedded in another noun then the pronouns can act independently of each other. Additionally, when we compare the available readings of the elided sentence in example 349 with the available readings of the elided sentence in 350, we conclude that simple pronouns (unembedded pronouns) act differently than noun-embedded pronouns in multiple pronoun verb phrase ellipsis.

Now that we have observed that simple and embedded pronouns act differently, we reexamine examples 73 and 345. Each example contains a trigger sentence with two pronouns whose antecedent is the syntactic subject. In contrast to example 350, only one of the pronouns is embedded in a noun phrase in examples 73 and 345. Using our approach, let's look at the representations of trigger sentences that generate *bad* meanings of the elided sentence to see if a pattern emerges. Consider, first, example 73. Recall that Sag's multiple pronoun constraint correctly predicts the available readings of the elided sentence in this example.

Example 73

Bill$_i$ believed he$_i$ loved his$_i$ wife.
Harry$_j$ did too.
Meanings:
1. Harry believed that Harry loved Harry's wife.
2. Harry believed that Bill loved Bill's wife.
3. Harry believed that Harry loved Bill's wife. (marginal)
4. *Harry believed that Bill loved Harry's wife.

The unacceptable reading of the elided sentence for this example is derived from the representation of the trigger sentence where the function representing *he* is equated with the function representing the subject while the function representing *his* is equated with the subject's lambda variable. This representation is shown in example 351.

Example 351

```
((def1) | (name (def1) Bill)),
   λ(x)(believe x [(and (he2 x),
                        λ(y)(love y ((def3 x y) |
                                     (and (wife (def3 x y))
                                          (possess (his4 x y)
                                                   (def3 x y))
                                          (= (his4 x y) x))))
                   (= (he2 x) (def1)))])
```

The bad meaning of the elided sentence is generated if we equate an embedded pronoun function with the subject's lambda variable and equate a simple pronoun function with another value for the subject in the trigger's representation. Given this description of a *bad* trigger, consider example 345. This example is similar to example 73, except the order of the pronouns is different.

Example 345

Bill$_i$ believed that his$_i$ wife loved him$_i$.
Harry$_j$ did too.
Meanings:
1. Harry believed that Harry's wife loved Harry.
2. Harry believed that Bill's wife loved Bill.
3. *Harry believed that Harry's wife loved Bill.
4. Harry believed that Bill's wife loved Harry. (marginal)

Sag's constraint predicts that the fourth reading of 345 should be incorrect. In fact, the incorrect reading of the elided sentence is the third reading. In our approach, the bad reading of the elided sentence is derived from the representation of the trigger sentence where the function representing *him* is equated with the function representing the subject and the function representing *his* is equated with the subject's lambda variable (shown in 352).

Example 352

```
((def1) | (name (def1) Bill)),
   λ(x)(believe x [((def2 x) |
                    (and (wife (def2 x))
                         (possess (his3 x) (def2 x))
                         (= (his3 x) x))),
                    λ(y)(and (love y (him2 x y))
                             (= (him2 x y) (def1)))])
```

Again, the bad reading of the elided sentence is generated if we equate the embedded pronoun function with the subject's lambda variable and equate the simple pronoun function with another value for the subject in the trigger's representation. Notice that this description of the bad trigger sentence in example 345 matches the description for the bad trigger in example 73. Each bad trigger sentence contains a simple pronoun equated with the subject's function and an embedded pronoun equated with the subject's lambda variable.

Now that we have described the types of trigger representations that generate *bad* meanings of an elided sentence, we consider those trigger representations that generate *good* meanings of the elided sentence. We hope to discover why the *good* trigger sentence representations give rise to more reasonable readings of the elided sentence than the *bad* ones. Consider the *good* representations of the trigger sentence in 73 (shown in 353).

Example 353

```
a. ((def1) | (name (def1) Bill)),
      λ(x)(believe x [(and (he2 x),
                           λ(y)(love y ((def3 x y) |
                                        (and (wife (def3 x y))
                                             (possess (his4 x y)
                                                      (def3 x y))
                                             (= (his4 x y) x))))
                           (= (he2 x) x))])
```

Example 353 continued:

```
b. ((def1) | (name (def1) Bill)),
      λ(x)(believe x [(and (he2 x),
                           λ(y)(love y ((def3 x y) |
                                        (and (wife (def3 x y))
                                             (possess (his4 x y)
                                                      (def3 x y))
                                             (= (his4 x y) (def1)))))
                           (= (he2 x) (def1)))])
c. ((def1) | (name (def1) Bill)),
      λ(x)(believe x [(and (he2 x),
                           λ(y)(love y ((def3 x y) |
                                        (and (wife (def3 x y))
                                             (possess (his4 x y)
                                                      (def3 x y))
                                             (= (his4 x y) (def1)))))
                           (= (he2 x) x))])
```

The trigger verb phrases in 353a and 353b are used to generate reasonable meanings of the elided sentence. Notice that in each case, the two pronoun functions are equated with the same representation for the subject. On the other hand, like the representation in 351, the other *good* trigger representation (shown in 353c) contains two pronoun functions equated with different values. However, unlike the representation in 351, the embedded pronoun function (his_4 x y) is equated with the subject's function (a function with no arguments) while the simple pronoun is equated with the subject's lambda variable. Notice that because of the meaning assigned to the embedded pronoun function, the function containing the pronoun function (i.e., (def_3 x y)) can be reduced (using the argument reduction constraint) to a function with no arguments (or a constant). In the *bad* representation of the trigger (shown in 351) the function containing the embedded pronoun function cannot be reduced to a constant.

Now consider the representation of the trigger sentences that generate *good* meanings of the elided sentence in 345.

Example 354

```
a. ((def1) | (name (def1) Bill)),
      λ(x)(believe x [((def2 x) |
                       (and (wife (def2 x))
                            (possess (his3 x) (def2 x))
                            (= (his3 x) x))),
                       λ(y)(and (love y (him2 x y))
                                (= (him2 x y) x))])
b. ((def1) | (name (def1) Bill)),
      λ(x)(believe x [((def2 x) |
                       (and (wife (def2 x))
                            (possess (his3 x) (def2 x))
                            (= (his3 x) (def1)))),
                       λ(y)(and (love y (him2 x y))
                                (= (him2 x y) (def1)))])
```

Example 354 continued:

```
c. ((def1) | (name (def1) Bill)),
      λ(x)(believe x [((def2 x) |
                          (and (wife (def2 x))
                               (possess (his3 x) (def2 x))
                               (= (his3 x) (def1)))),
                          λ(y)(and (love y (him2 x y))
                                   (= (him2 x y) x))])
```

The trigger representations in 354a and 354b contain two pronoun functions equated with the same values. Again, both of these representations are used to generate reasonable meanings of the elided sentence. On the other hand, the representation in 354c contains two pronoun functions equated with different values corresponding to the subject. Again, the function containing the embedded pronoun function (i.e., (def_2 x)) can be reduced to a function with no arguments. In contrast, in 352, (def_2 x) cannot be reduced at all.

Together examples 73, 345, 350, and 349 suggest that two pronouns that refer to the same noun phrase in a trigger sentence must refer to that noun phrase in the same way unless one of the pronouns is contained in a noun phrase that can be replaced by a constant. A definite noun phrase, represented as a definite function with a restriction, can be replaced by a constant if its restriction is closed. For a restriction to be closed, all of the pronouns in the restriction must be equated with functions with closed restrictions or with discourse entities. In fact, using argument reduction, a definite function is replaced by a function with no arguments when its restriction contains no unbound variables (given that equality is substitution). Since we derive the meaning of an elided sentence from a completely disambiguated trigger sentence, this rule can be used to filter out representations of trigger sentences that give rise to impossible meanings for elided sentences.

Our pronoun rule works quite well for all of the previous examples. However, in each of the past examples, if a pronoun was embedded, it was embedded in a definite noun phrase (and that noun phrase contained no other embedded noun phrases). It is reasonable for a definite function to be replaced by a function with no arguments if all of its embedded pronouns are replaced by functions with no arguments. However, what happens when the container of an embedded pronoun is an indefinite? Consider example 355.

Example 355

```
George_i believes he_i likes a friend of his_i.
Fred_j does too.
meanings:
1. Fred believes that George likes a friend of George's (the same one).
2. Fred believes that George likes a friend of George's (a different one).
3. Fred believes that Fred likes a friend of Fred's.
4. Fred believes that Fred likes a friend of George's (the same one).
5. Fred believes that Fred likes a friend of George's (a different one).
6. *Fred believes that George likes a friend of Fred's.
```

Notice that there are more readings of the elided sentence when a pronoun is embedded in an indefinite noun phrase, than when a pronoun is embedded in a definite. There are six representations for the trigger sentence in 355. In each representation, the indefinite is represented as a function of those variables associated with operators that have scope over the indefinite (taking into account those operators that must have scope over the indefinite in order to bind a variable in

its restriction).

Example 356

```
a. ((def1) | (name (def1) George)),
      λ(x)(believe x [(and (he2 x),
                           λ(y)(like y ((indef3) |
                                        (and (friend (indef3))
                                             (possess (his4 x y)
                                                      (indef3))
                                             (= (his4 x y) (def1)))))
                          (= (he2 x) (def1)))])
   ; both pronouns equal (def1)
   ; and the lambda operators do not have scope over the existential
b. ((def1) | (name (def1) George)),
      λ(x)(believe x [(and (he2 x),
                           λ(y)(like y ((indef3 x) |
                                        (and (friend (indef3 x))
                                             (possess (his4 x y)
                                                      (indef3 x))
                                             (= (his4 x y) (def1)))))
                          (= (he2 x) (def1)))])
   ; both pronouns equal (def1)
   ; and λ(x) has scope over the existential
c. ((def1) | (name (def1) George)),
      λ(x)(believe x [(and (he2 x),
                           λ(y)(like y ((indef3 x) |
                                        (and (friend (indef3 x))
                                             (possess (his4 x y)
                                                      (indef3 x))
                                             (= (his4 x y) x))))
                          (= (he2 x) x))])
   ; both pronouns equal x
   ; and λ(x) must have scope over the existential
d. ((def1) | (name (def1) George)),
      λ(x)(believe x [(and (he2 x),
                           λ(y)(like y ((indef3) |
                                        (and (friend (indef3))
                                             (possess (his4 x y)
                                                      (indef3))
                                             (= (his4 x y) (def1)))))
                          (= (he2 x) x))])
   ; the embedded pronoun equals (def1), the simple pronoun equals x
   ; and the lambda operators do not have scope over the existential
```

Example 356 continued:

```
e. ((def1) | (name (def1) George)),
      λ(x)(believe x [(and (he2 x),
                           λ(y)(like y ((indef3 x) |
                                        (and (friend (indef3 x))
                                             (possess (his4 x y)
                                                      (indef3 x))
                                             (= (his4 x y) (def1)))))
                           (= (he2 x) x))])
   ; the embedded pronoun equals (def1), the simple pronoun equals  x
   ; and λ(x) has scope over the existential
f. ((def1) | (name (def1) George)),
      λ(x)(believe x [(and (he2 x),
                           λ(y)(like y ((indef3 x) |
                                        (and (friend (indef3 x))
                                             (possess (his4 x y)
                                                      (indef3 x))
                                             (= (his4 x y) x))))
                           (= (he2 x) (def1)))])
   ; The simple pronoun equals (def1), the embedded pronoun equals x
   ; and λ(x) must have scope over the existential
```

We cannot account for this example by using our current pronoun rule. The only bad reading of the elided sentence in 355 is the sixth meaning (derived from 356f). However, our pronoun rule eliminates the representations of the trigger sentence in 356e and 356f. In both cases, the two pronouns refer to their antecedents in different ways and the embedded pronoun is not embedded in a function with no arguments. Hence, the fifth and sixth readings of the elided sentence in 355 are ruled out. Our pronoun rule must change if we are to handle this example.

Consider another example in which the pronoun is embedded in a noun phrase, but this time the noun phrase is universally quantified.

Example 357

```
George_i believes every girl he knows likes him_i.
Fred_j does too.
meaning:
1. Fred believes that every girl George knows likes George.
2. Fred believes that every girl Fred knows likes Fred.
3. Fred believes that every girl George knows likes Fred.
4. *Fred believes that every girl Fred knows likes George.
```

Notice that the third reading in 357, a mixed reading, is reasonable despite the fact that we cannot represent the universal containing the embedded pronoun as a function with no arguments. The representations for the trigger sentence from example 357 are shown in 358.

Example 358

```
a. ((def1) | (name (def1) George)),
      λ(x)(believe x [∀y: (and (girl y)
                               (he3 x), λ(z)(know z y)
                               (= (he3 x) (def1)))
                          y, λ(w)(and (love w (him2 x w))
                                      (= (him2 x w) (def1)))])
   ; both pronouns equal (def1)
b. ((def1) | (name (def1) George)),
      λ(x)(believe x [∀y: (and (girl y)
                               (he3 x), λ(z)(know z y)
                               (= (he3 x) x))
                          y, λ(w)(and (love w (him2 x w))
                                      (= (him2 x w) x))])
   ; both pronouns equal x
c. ((def1) | (name (def1) George)),
      λ(x)(believe x [∀y: (and (girl y)
                               (he3 x), λ(z)(know z y)
                               (= (he3 x) (def1)))
                          y, λ(w)(and (love w (him2 x w))
                                      (= (him2 x w) x))])
   ; the simple pronoun equals x, the embedded pronoun equals (def1)
d. ((def1) | (name (def1) George)),
      λ(x)(believe x [∀y: (and (girl y)
                               (he3 x), λ(z)(know z y)
                               (= (he3 x) x))
                          y, λ(w)(and (love w (him2 x w))
                                      (= (him2 x w) (def1)))])
   ; the embedded pronoun equals x, the simple pronoun equals (def1)
```

Our current pronoun rule eliminates the trigger representations in 358c and 358d. However, the third reading of the elided sentence in 357 is reasonable. Hence, our pronoun rule should be modified to allow the trigger representation in 358c, while eliminating 358d.

While it is true that our pronoun rule as it currently stands handles examples 73, 345, 347, 348, 349, and 350, it fails to handle examples 355 and 357. When a single pronoun is embedded in a definite noun phrase that can be replaced by a function with no arguments, the remaining pronoun can act independently of the embedded pronoun. However, once we consider definites together with indefinites and universals, a different picture emerges. All of the representations of trigger sentences that generate bad readings for the elided sentence (i.e., 351, 352, 356f, and 358d) have one thing in common. The two pronoun functions are equated with different values, and the restriction of the noun phrase containing one of the pronouns is not closed (because each embedded pronoun function is equated with the subject's lambda variable).

Does this mean that when a pronoun is embedded in a noun phrase with a closed restriction, then the other pronoun can act independently? Example 359 suggests that restriction closure is too strict a requirement for allowing two pronouns to act independently.

Example 359

Fred$_i$ told every dog breeder$_j$ that the dog she$_j$ gave him$_i$ bit him$_i$.
George$_k$ did too.
Meanings:
1. George told every dog breeder that the dog she gave George bit George.
2. George told every dog breeder that the dog she gave Fred bit Fred.
3. George told every dog breeder that the dog she gave Fred bit George.
4. *George told every dog breeder that the dog she gave George bit Fred.

Consider the four trigger representations (shown in 360) that give rise to the four meanings of the elided sentence.

Example 360

```
a. ((def1) | (name (def1) Fred)),
      λ(x)(tell x [∀y: (dog-breeder y) y]
                  [((def2 x y)|
                      (and (dog (def2 x y))
                           (she3 x y), λ(z)(and (give z (him4 x y z))
                                                (= (him4 x y z) x))
                           (= (she3 x y) y)))
                      λ(w)(and (bite w (him5 x y w))
                               (= (him5 x y w) x))])
b. ((def1) | (name (def1) Fred)),
      λ(x)(tell x [∀y: (dog-breeder y) y]
                  [((def2 x y)|
                      (and (dog (def2 x y))
                           (she3 x y), λ(z)(and (give z (him4 x y z))
                                                (= (him4 x y z) (def1)))
                           (= (she3 x y) y)))
                      λ(w)(and (bite w (him5 x y w))
                               (= (him5 x y w) (def1)))])
c. ((def1) | (name (def1) Fred)),
      λ(x)(tell x [∀y: (dog-breeder y) y]
                  [((def2 x y)|
                      (and (dog (def2 x y))
                           (she3 x y), λ(z)(and (give z (him4 x y z))
                                                (= (him4 x y z) (def1)))
                           (= (she3 x y) y)))
                      λ(w)(and (bite w (him5 x y w))
                               (= (him5 x y w) x))])
d. ((def1) | (name (def1) Fred)),
      λ(x)(tell x [∀y: (dog-breeder y) y]
                  [((def2 x y)|
                      (and (dog (def2 x y))
                           (she3 x y), λ(z)(and (give z (him4 x y z))
                                                (= (him4 x y z) x))
                           (= (she3 x y) y)))
                      λ(w)(and (bite w (him5 x y w))
                               (= (him5 x y w) (def1)))])
```

The trigger verb phrases in 360a, 360b, and 360c can be used to generate *good* readings of the elided sentence. The trigger representations in 360a and 360b are fine since both pronoun functions

are equated with the same value for the subject. In contrast, the representations in 360c and 360d contain two pronoun functions, each equated with a different subject value. Also notice that the pronoun function (him_4 x y z) in 360c and 360d is embedded in a noun phrase whose restriction is not closed. In both restrictions, (she_3 x y) is equated with a universal variable. However, the representation of the trigger in 360c can be used to generate a *good* reading of the elided sentence, while the representation in 360d cannot. We can explain this difference, however. Notice the restriction in 360c does not change when a new subject is applied to the trigger verb phrase (to provide a meaning for the elided sentence), while the restriction in 360d does. Hence, we suggest that when two pronouns in a trigger verb phrase refer to the same subject, they must refer to that subject in the same way, unless one of them is embedded in a restriction which does not change when a new subject is applied to the lambda function representing the trigger verb phrase. We call a restriction that does not change when a new subject is applied to the verb phrase ***descriptively closed.*** Hence, when one of the pronouns is embedded in a descriptively closed restriction, the other pronoun can vary freely[3].

Because a restriction on a noun phrase is a proposition, we could possibly claim that whenever a pronoun is embedded in a descriptively closed proposition, then the other pronoun whose antecedent is the same subject can act independently. Hence, we suggest the following constraint to filter trigger sentences used to generate the meaning of an elided sentence.

Multiple Pronoun Constraint:

> If two pronouns within a trigger verb phrase refer to the same surface subject, then they must always refer to that subject (in the same way in the logical form for the trigger sentence), unless one of the pronouns is embedded in a descriptively closed proposition, then the remaining pronoun can vary freely (that is it can equal the variable or the function associated with its antecedent).

This constraint can be extended to sentences containing more than two pronouns that refer to the syntactic subject. All of the pronouns in a trigger sentence that refer to the same subject must refer to that subject in the same way, unless some are no longer considered because they are contained in descriptively closed restrictions.

Our multiple pronoun constraint works quite well for all of the examples that we have considered, but there is one other type of example we should consider. In this type of example, the trigger sentence contains two simple pronouns that refer to the matrix subject of the sentence. However, one of the pronouns is embedded in a clause contained in the matrix sentence and in a different

---

[3]Notice that issues of control often require a pronoun to be equated with a pronoun instead of that pronoun's antecedent. Consider example 361.

Example 361
Fred$_i$ believed that he$_i$ loved himself$_i$.
George$_k$ did too.
Meanings:
1. George believed that Fred loved Fred.
2. George believed that George loved George.
3. *George believed that Fred loved George.
4. *George believed that George loved Fred.

Cases like this should work out fine without the multiple pronoun constraint. The antecedent for *himself* must be *he*. We obtain the correct readings of the elided sentence without the multiple pronoun constraint because the meaning assigned to *he* is inherited by *himself*.

clause than the other pronoun[4]. Example 362 is this type of example.

Example 362

Fred$_i$ said that he$_i$ believes he$_j$ will win.
George$_j$ did too.
Meanings:
1. George said that George believes George will win.
2. George said that Fred believes Fred will win.
3. George said that George believes Fred will win.
4. *George said that Fred believes that George will win.

Despite the fact that the two pronouns in the trigger sentence are not embedded in noun phrases, one of them is embedded in a sentence. Hence, our constraint correctly predicts that three readings for the elided sentence should be available. Based on our constraint, the elided sentence of 362 should receive three different meanings (it has three, as the reader can verify). The interesting thing about this example is that the object of *believes* is a sentence, or in our representation a proposition (i.e., a lambda function plus all of the terms applied to that function to make it a proposition). Like a restriction, a sentence is a proposition (and it can be descriptively closed). To see how our constraint handles example 362, consider the representation of the trigger sentence used to generate the meanings of the elided sentence (shown in 363).

Example 363

```
a. ((def3) | (name (def3) Fred)),
      λ(x)(say x [(and (he1 x),
                        λ(y)(believe y [(and (he2 x y), λ(z)(win z)
                                             (= (he2 x y) x))])
                       (= (he1 x) x))])
   ; both pronouns equal x
b. ((def3) | (name (def3) Fred)),
      λ(x)(say x [(and (he1 x),
                        λ(y)(believe y [(and (he2 x y), λ(z)(win z)
                                             (= (he2 x y) (def3)))])
                       (= (he1 x) (def3)))])
   ; both pronouns equal (def3)
c. ((def3) | (name (def3) Fred)),
      λ(x)(say x [(and (he1 x),
                        λ(y)(believe y [(and (he2 x y), λ(z)(win z)
                                             (= (he2 x y) (def3)))])
                       (= (he1 x) x))])
   ; the proposition-embedded pronoun equals (def3), the other equals x
d. ((def3) | (name (def3) Fred)),
      λ(x)(say x [(and (he1 x),
                        λ(y)(believe y [(and (he2 x y), λ(z)(win z)
                                             (= (he2 x y) x))])
                       (= (he1 x) (def3)))])
   ; the proposition-embedded pronoun equals x, the other equals (def3)
```

The representations in 363a and 363b generate reasonable meanings of the elided sentence. In both cases, the pronouns are equated with the same values. On the other hand, in both 363c and

[4]Notice that example 349 is not an example of this type.

363d, the pronouns are equated with different values. However, in 363c, (he_2 x y) is embedded in a sentence which is descriptively closed with respect to the lambda variable x (after substitutions for equality statements are made). Because the proposition in 363c associated with ***he will win*** is descriptively closed, the remaining pronoun is free to refer to the subject in another way. Hence, the trigger representation generates a reasonable meaning for the elided sentence. In contrast, the trigger representation in 363d generates a ***bad*** meaning for the elided sentence. Notice that the proposition containing (he_2 x y) is not descriptively closed. By allowing descriptively closed propositions to act as a barrier between the embedded pronoun and another pronoun outside of the proposition, we can account for all of the examples in this chapter, including example 362.

In this chapter, we have suggested a mechanism to rule out unlikely meanings of elided sentences given that its trigger sentence contains two or more pronouns that refer to the syntactic subject of that sentence. This constraint is based on the observation that pronouns that refer to a syntactic subject in a trigger sentence must refer in the same way to that subject, unless one of the pronouns is embedded in the descriptively closed proposition (i.e., either the restriction of a noun phrase or a sentence). We provide no justification for this constraint except that it fits the data examined. However, the constraint was incorporated into our implementation described in Chapter six. All of the examples discussed in this chapter are covered by our implementation of the constraint (many of these examples are also listed in Appendix A).

Most of the examples in this chapter were judged independently by ten informants. In general, the bad mixed readings were considered bad by all. However, not all subjects judged the ***good*** mixed readings as reasonable. Many required that all pronouns with the same antecedent must refer to it in the same way. However, about half of the informants were able to understand the ***good*** mixed readings of the elided sentence. We should also point out that people did not enjoy judging the acceptability of the readings available in multiple pronoun verb phrase ellipsis. We had them judge the meanings of elided sentences whose trigger contained two or three pronouns. The case with three seemed harder for all, not surprisingly.

Chapter 6

Implementation

6.1 Introduction

In this chapter, we describe the implementation of a system that generates the logical form for a sentence. The parser takes a sentence as input and outputs its logical form. Additionally, our system provides the meaning for a sentence with verb phrase ellipsis. Once the logical form for a sentence is available, the system examines it for verb phrase ellipsis. If there is no verb phrase ellipsis in the logical form for a sentence, it is placed on a stack of possible trigger sentences. On the other hand, if it contains a verb phrase ellipsis, its trigger sentence is located and disambiguated so that the meaning of the elided sentence can be provided. In this thesis, we do not provide a way to determine antecedents for pronouns and anaphoric definites or trigger verb phrases for elided verb phrases. We rely on the user of the program to handle this aspect of sentence comprehension. However, our initial representations for pronouns and definites are used by the program to limit the pool of possible intrasentential antecedents.

This implementation demonstrates three things. First, the logical form for a sentence can be generated using only information provided by the parse tree. In fact, the logical form is generated by a parser. Hence, when the parse tree is available, so is its logical form. Second, the implementation demonstrates that the corpus of examples listed in Appendix A can be handled using the representations devised for pronouns, definite noun phrases, and indefinite noun phrases. In fact, the implementation demonstrates, not only the adequacy of these representations, but also their usefulness in limiting what the antecedents for pronouns and anaphoric definites can be. Finally, the implementation demonstrates that an intermediate representation for a trigger sentence can be updated to a reasonable final meaning for that sentence (and consequently, a reasonable meaning for an elided sentence dependent on that trigger for its meaning).

In the remaining sections of this chapter, we describe our system, which generates logical forms for sentences as well as meanings for sentences with verb phrase ellipsis. First, we describe routines that are necessary to provide the logical form for a sentence.

6.2 Generating Logical Form in a Parser

We modified a parser to output the logical form for a sentence, in addition to its parse tree. In compliance with the modularity constraint from Chapter one, the parser uses only syntactic and

sentence-level information to generate logical form. In this section, we briefly describe the routines we added to the grammar of a parser to generate the logical form for a sentence. These routines provide representations for all of the sentences in Appendix A. In the previous chapters, we assumed that a modification of Reinhart's c-command rule should be used to determine those quantified noun phrases that can affect the meaning of a pronoun or definite noun phrase. In this chapter, because we do not discuss any sentences with preposed phrases (e.g., *In his room, each child seeks solitude*), we can use a mechanism which achieves the same results as the c-command rule does, but which allows us to construct the representations for pronouns as we parse a sentence from the left to the right. To determine the argument lists for pronoun functions or definite functions, we make use of an active variable list. Whenever a quantified noun phrase is processed in a sentence, its variable is added to the active variable list. Also, when a subject is processed, its lambda variable is added to the active variable list. Additionally, if the subject is quantified, its variable is removed from the active variable list when its lambda variable is added. Later when a sentence is complete, the the lambda variable for the subject of the sentence and the variables introduced by the quantified noun phrases in the sentence are removed from the active variable list.

Three types of routines are necessary to provide the logical form for a sentence. The first type of routine is responsible for creating the structures in our representation for a sentence. These include routines called at the beginning of a sentence as well as routines called when determiners, pronouns, proper nouns, possessive adjuncts, and prepositional phrases are processed. We also include the routine called when a sentence has been completely processed (though it does not fit well into this category). These routines are summarized below:

1. **Begin a Sentence:**
   When the beginning of a sentence is detected, create a structure for a sentence consisting of a subject slot, a slot for the lambda operator and variable, and a slot for the predicate-argument structure. The representation of the verb phrase consists of the lambda operator (and its variable) plus the predicate-argument structure (as discussed in Chapter two). Initially, no information is indicated in the sentence structure.

2. **Determiner:**
   Depending on the type of the determiner, create a structure of the appropriate type.

   If the determiner is *a*, *some*, or *an*, then create a structure consisting of an existential quantifier, a variable, and a restriction (initially the restriction is empty). This representation for indefinites is discussed in Chapter four. When the representation for the indefinite is created, add its variable to the active variable list.

   If the determiner is *every* or *each*, then create a structure consisting of a universal quantifier, a variable, and a restriction (initially the restriction is empty). When the representation for the universal noun phrase is created, add its variable to the active variable list.

   If the determiner is *the*, then create a definite function (discussed in Chapter three) with a unique name and an argument list consisting of the variables on the active variable list. However, the argument list of the function is incomplete until the parser processes the entire noun phrase. Add any variables corresponding to embedded quantified noun phrases not subject to the complex noun phrase constraint to the argument list of the function. Initially, the restriction of the definite function is empty.

3. **Pronoun:**
   If a pronoun is processed, create a pronoun function with a unique name and an argument

list consisting of the variables on the active variable list. We also keep track of whether the pronoun is reflexive or not.

4. **Proper Noun:**
   If a proper noun is processed, create a definite function with a unique name and an argument list consisting of the variables on the active variable list. Additionally, add information about the name of the individual denoted by the proper noun to the restriction of the function.

5. **Possessive Adjunct:**
   When the noun following a possessive noun phrase (called a possessive adjunct) is processed, create a definite function (the possessive noun phrase indicates that the parser is processing a definite noun phrase) with a unique name and an argument list consisting of the variables on the active variable list. The possessive adjunct provides information about the type of entity described by the the definite function. Add this information to the restriction of the function. For example, if the noun phrase *every man's mother* is being parsed, and the parser detects that *mother* is a possessive adjunct, then it creates a definite function for the possessive noun phrase, say (def_3 x y) (assuming that *every man* is represented as $\forall x$: (man x) x and the function is created in the scope of the lambda operator $\lambda(y)$). In this case, the parser adds the proposition (mother (def_3 x y)) to the restriction of that function. The parser also adds information indicating the possessive relationship between the possessive noun and the possessive adjunct. So for our example, the parser adds (possess x (def_3 x y)) to the restriction of the function (def_3 x y).

6. **Prepositional Phrase:**
   When a prepositional phrase is parsed, create a structure for a prepositional phrase (i.e., a list consisting of the preposition and the representation for the object of the preposition).

7. **End a Sentence:**
   When a sentence is complete, remove any variables introduced in that sentence from the active variable list. These include the variables corresponding to quantified noun phrases parsed in that sentence plus the lambda variable introduced by the subject of that sentence. When the sentence is ended, those variables cannot affect the representation of future noun phrases.

Another type of routine is responsible for adding information to a structure created by one of the routines mentioned above. Sometimes the type of word processed by the parser triggers this type of logical form routine. For example, if the parser encounters an adjective, it adds information about that adjective to the restriction of the current noun phrase (i.e., the noun phrase the parser is processing). Information is also added to the representation of a sentence when the parser recognizes that a constituent is the subject, object, or indirect object of the sentence. We also include attachment routines in this section (i.e., relative clause attachment and prepositional phrase attachment). The routines to handle common nouns, adjectives, verbs, particles, relative clause attachment, prepositional phrase attachment, subjects, objects, and indirect objects are summarized next.

1. **Common Noun:**
   When a common noun is processed, add information to the restriction of the current noun phrase indicating its type. For example, if the parser processes the word *the* in the noun phrase *the boy*, then it creates a definite function (e.g., (def_{55} x y)). Later when the parser processes the word *boy*, it adds the proposition (boy (def_{55} x y)) to the function's restriction

2. **Adjective:**
   If an adjective is processed and it is part of the current noun phrase, then add information about that adjective to the restriction for the current noun phrase (i.e., the individual denoted by the current noun phrase has the property indicated by the current adjective). For example, assume that the parser is processing the noun phrase ***the big block.*** Also assume that the parser creates the function (def_{79} y z) when it parses the word ***the.*** Then, when the adjective ***big*** is processed, the proposition (big (def_{79} y z)) is added to the restriction of the function (def_{79} y z). On the other hand, if the adjective is a predicate adjective (e.g., ***happy*** in ***Fred is happy***), then make the adjective the predicate in the predicate-argument structure for the verb phrase.

3. **Verb:**
   When a verb is processed, make the verb the predicate in the predicate-argument structure for the verb phrase.

4. **Particle:**
   When a particle is processed, join it with the predicate of the current clause. For example if the predicate is ***look*** and the particle is ***up***, make a new predicate ***look-up*** to replace the old predicate.

5. **Prepositional Phrase Attachment:**
   If the current prepositional phrase is a predicate prepositional phrase (e.g., *He is in the water*), make the preposition the predicate in the predicate-argument structure for the clause. Add a list consisting of the word *object* and the representation of the object of the preposition to the predicate-argument structure. If the prepositional phrase should be attached to the current noun phrase, then add the representation of the prepositional phrase to the restriction for that noun phrase. If the prepositional phrase should be attached to the verb phrase, add the representation of the prepositional phrase to the predicate-argument structure used to represent the verb phrase.

6. **Relative Clause Attachment:**
   Once a relative clause has been processed, add its representation to the restriction for the current noun phrase.

7. **Subject:**
   When the subject of a sentence is detected, place the representation for the subject in the subject slot of the the sentence, and create a lambda variable corresponding to the subject. If the subject is quantified, remove its variable from the active variable list and place its lambda variable in the list instead. Add information to the predicate-argument structure of the sentence concerning the logical role of the subject. Because it is easier to distinguish arguments of a predicate-argument structure by using slot-filler notation rather than positional notation (i.e., one in which the order of the arguments indicates their role), we use slot-filler notation to indicate the logical roles of noun phrases. If a sentence has active voice, add a list consisting of the word *subject* and the subject's lambda variable to the predicate-argument structure for the sentence. On the other hand, if it has passive voice, then add a list consisting of the word *object* and the subject's lambda variable instead.

8. **Object:**
   When the object of a sentence is detected, add a list consisting of the word *object* and the representation of the object to the predicate-argument structure for the verb phrase.

9. **Indirect Object:**
   When the indirect object of a sentence is detected, Add a list consisting of the word ***indirect-object*** and the representation of the indirect object to the predicate-argument structure for the verb phrase.

Finally, the parser invokes a set of routines to handle a variety of linguistic phenomena including wh-traces, verb phrase ellipsis, subject equi, object equi, subject lowering, and object lowering[1]. Next, we briefly summarize each routine.

1. **Verb Phrase Ellipsis:**
   When verb phrase ellipsis is detected, create a dummy predicate for verb phrase representation. This routine keys off a verb phrase consisting of an auxiliary not followed by a verb (or a noun phrase).

2. **Wh-word:**
   If a wh-word is processed and it occurs in a relative clause, use the representation of the relative clause's head (i.e., either the variable or the function without its restriction) as the representation for the wh-word.

3. **Subject Equi:**
   Subject equi (or subject controlled equi) is a linguistic phenomenon in which the subject of the matrix sentence also acts as the subject of an embedded verb phrase. For example, in the sentence *Mary wants to finish*, *Mary* seems to fill the role of subject in the matrix sentence and the role of subject for the verb phrase *to finish*. If the parser detects a subject equi, then it makes a copy of the subject without its restriction to represent the missing subject of the embedded verb phrase.

4. **Object Equi:**
   Object equi (or object controlled equi) is similar to subject equi except that the object of the matrix sentence also acts as the subject of an embedded verb phrase. For example, in the sentence *Mary persuaded Paul to go*, *Paul* is both the object of the matrix sentence and the subject of *to go*. If the parser detects an object equi, then it makes a copy of the object without its restriction to represent the missing subject of the embedded verb phrase.

5. **Subject Lowering:**
   Subject lowering (or raising to subject) appears to be similar to subject equi. However, unlike subject equi, in subject lowering, the subject of the matrix sentence fills no logical role in the predicate-argument structure corresponding to the matrix sentence. For example, in the sentence *Mary seemed to be happy*, *Mary* fills the role of logical subject in the predicate-argument structure corresponding to *to be happy* but does not fill a role in the predicate-argument structure corresponding to the verb *seemed*. In our system, when the parser detects subject lowering, it abstracts the surface subject from the predicate-argument structure of the verb marked for subject lowering and adds its logical role to the predicate-argument structure of the embedded sentence. This is necessary to handle verb phrase ellipsis. Consider example 364.

[1] In using these terms, we are not committing to a transformational grammar approach. We use the terms simply to indicate the type of syntactic phenomenon encountered.

Example 364

```
Mary seemed to be happy.
George did too.
Meanings:
George seemed to be happy.
```

Because the verb *seem* takes a sentence as its object, the trigger sentence is usually represented as (seem (happy Mary)). However, to handle example 364, we must abstract the subject from the verb phrase corresponding to *seem* even though it fills no logical role in that predicate-argument structure. Hence, the trigger sentence is initially represented as shown in 365.

Example 365

```
Mary seemed to be happy.

((def88) | (name (def88) Mary)),
   λ(x)(seem [(def88), λ(y)(happy (subject y))])
```

Notice that the subject is abstracted from the predicate-argument structure for *seem* (i.e., the verb marked for subject lowering) though it fills a logical role in the embedded predicate-argument structure (i.e., the predicate-argument structure for ***happy***).

6. **Object Lowering:**
Object lowering (or raising to object) is similar to subject lowering except the object is involved. For example, in the sentence *Someone wanted every student to talk*, *every student* fills no logical role in the predicate-argument structure corresponding to the verb *wanted* though it fills the role of logical subject in the predicate-argument structure corresponding to *to talk*. If the parser detects object lowering, it lowers the representation of the *object* from object position in the top-level sentence to subject position in the embedded sentence. For example, the sentence *Someone wanted every student to talk* is represented as follows:

Example 366

```
Someone wanted every student to talk.

∃(x): (person x)
 x, λ(y)(want (subject y) [∀z: (student z)
                            z, λ(w)(talk (subject w))])
```

If the lowered object is quantified, it could have scope over noun phrases from its original level. In example 366, *every student* could have scope over *someone*. We can detect this scope ambiguity because the quantified variable z is added to the active variable list when the parser is processing the matrix sentence.

The logical form routines summarized in this section are called by our parser as it processes a

sentence. When the parser processes components of a sentence, it adds information to the representation of the sentence. Once the end of a sentence is reached, the logical form for the sentence is available. However, because the parser provides all parses for a sentence and because logical form depends on the parse tree, the parser can sometimes generate multiple logical forms for a sentence. In such a case, the user is asked to pick the intended parse (and hence, the intended logical form). Logical form is created in a nearly compositional way. However, because the program must keep track of variables associated with quantified noun phrases in order to provide initial representations for pronouns and definites, complete compositionality is impossible.

6.3 Updating Logical Form with User Intervention

In this section, we describe the second part of our program, that is, the part that provides meanings for sentences with verb phrase ellipsis. The input for this part of the program is the logical form for a sentence, output by the parser. The program examines the logical form to see if it contains verb phrase ellipsis. If not, then the logical form is placed on the stack of recently parsed sentences, with no additional processing. However, if the sentence is elided, then the program searches for the logical form of the trigger sentence. One might assume that the trigger sentence is always the immediately preceding sentence, but this does not always hold.

We assume that the ambiguity in an elided sentence's meaning arises because there are many ways to indicate the final meaning of its trigger sentence. Before the program can determine the meaning of an elided sentence, it must locate the trigger sentence and determine that sentence's meaning. To determine a trigger sentence's meaning, the program must find antecedents for all pronouns and anaphoric definite noun phrases, determine quantifier scoping, and pinpoint meanings of all non-anaphoric definites. In the rest of this section, we describe routines to handle anaphora (in section 6.3.1), routines that determine the necessary arguments of a definite function and replace that function with a function of the necessary arguments (in section 6.3.2), and routines to handle quantifier scoping (in section 6.3.2). Additionally, we briefly discuss the routines that assign meaning to an elided verb phrase (in section 6.3.3).

6.3.1 Anaphora Update

In this section, we discuss the routines which determine (with user intervention) antecedents for pronouns and anaphoric definites. These routines demonstrate the usefulness of the functional representation of pronouns and definites for limiting the pool of possible intrasentential antecedents.

Antecedents for pronouns and anaphoric definites are classified as either intrasentential or intersentential antecedents. Intersentential antecedents behave differently than intrasentential antecedents (discussed in Chapter two). Hence, when the program searches for the antecedent of an anaphoric noun phrase, it must be informed whether the antecedent is found within the same or a previous sentence. If the antecedent occurs in a previous sentence, then the user also provides a discourse entity for that antecedent. In verb phrase ellipsis, such anaphoric references can be modeled adequately using a constant. However, if the antecedent falls within the same sentence, then the program considers all noun phrases in the sentence as possible antecedents for the anaphora. Because an anaphoric noun phrase may be incompatible with some of the potential antecedents, the program checks each of the antecedent candidates to see if it is compatible with the anaphoric noun phrase. Because antecedents of definites have different properties than antecedents of pronouns, different

compatibility routines are provided for anaphoric definites and pronouns. Each is described in this section.

First, we consider the pronoun compatibility routine. This routine, which checks whether a noun phrase can be the intrasentential antecedent for a pronoun, is divided into two parts. The first part uses the type of pronoun to determine whether it is compatible with a set of possible antecedents. For example, if the pronoun is reflexive, its antecedent must occur in the same clause as the pronoun. Additionally, its antecedent cannot be embedded in another noun phrase. On the other hand, if the pronoun is not reflexive, its antecedent cannot be in the same clause as the pronoun unless the pronoun or its antecedent is embedded in another noun phrase. For example, the antecedent for the pronoun *him* in *Fred saw him* cannot be the same as for *himself* in *Fred saw himself*. Consider the initial representation for *Fred saw him*.

Example 367

```
Fred saw him.
```

```
((def7) | (name (def7) Fred)),
   λ(y)(see y (him2 y))
```

Despite the fact that it is legal to replace (him_2 y) with (def_7), the compatibility checker does not allow an anaphoric link between the two functions. The pronoun *him* is not reflexive and neither the pronoun nor *Fred* is embedded in another noun phrase (in contrast to *Fred's mother loves him* or *Fred saw the boy who knew him*). Additionally, a pronoun cannot typically be embedded in its antecedent. Consider example 368.

Example 368

```
Fred likes a friend of his.
```

The pronoun *his* cannot have *a friend of his* as its antecedent. There is one obvious exception. When a pronoun is embedded in a relative clause attached to a noun phrase, the wh-trace is used to determine the compatibility of a pronoun with its antecedent. For example:

Example 369

```
The boy who saw himself is happy.
```

The pronoun *himself* is embedded in its antecedent. However, the representation of the relative pronoun *who* is used in this case to determine compatibility. Since the pronoun is reflexive, it is compatible with the wh-trace. Each of the compatibility issues summarized above are concerns for all linguistic theories of anaphora.

To linguistic compatibility, we add an additional level of compatibility which is determined by examining the logical form for a pronoun and the logical form for its proposed antecedent. We claim that a pronoun function limits what its antecedent can be (as discussed in Chapter two). If we decide that the antecedent of a pronoun is a subject, and the pronoun refers to the subject indirectly, then unless the subject's lambda variable is an argument of the pronoun function, that pronoun cannot take that noun phrase as its antecedent. If the proposed antecedent is quantified and the variable is an argument of the pronoun function, then the pronoun can take the quantified noun phrase as its antecedent. On the other hand, if a universally quantified variable is not included in the argument list of the pronoun's function, then that quantified variable cannot be the assigned

value of the pronoun. Consider example 370.

Example 370

The man who saw every woman chased her.

```
((def47) | (and (man (def47))
                (def47), λ(x)(chase x [∀y: (woman y) y]))),
   λ(z)(chase z (her48 z))
```

Because *her* is represented as a function of *z*, *every woman* cannot be the antecedent for that pronoun. On the other hand, if the antecedent is an indefinite and the existential variable is not included in the argument list of the pronoun function, the pronoun function could still be compatible with the indefinite if the existential's variables are replaced by a function compatible with the pronoun function. If the proposed antecedent is a definite function, the pronoun function may be compatible with the initial definite function. However, it may not be compatible until after the meaning of the definite function is pinpointed (either after anaphora resolution or after argument reduction, depending on whether the definite is anaphoric or not). When more information is needed to determine compatibility, the program obtains the information first. This flexibility is necessary for handling the variety of examples discussed in this thesis. Finally, pronouns can be the antecedent for a pronoun given that the functions are compatible or given that the antecedent pronoun's value is compatible with the pronoun in question.

The second compatibility routine determines whether a definite can be anaphorically dependent on a noun phrase within a given sentence. Though definites can have intrasentential antecedents, compatibility is very limited. It seems that definites are only compatible with antecedents that do not c-command them (consider examples 242, 108, 147). If the antecedent is embedded in another sentence or embedded in a noun phrase in the same sentence, then the pronoun can be anaphorically dependent on it. However, an anaphoric definite cannot be embedded in its antecedent and we assume that a pronoun can never be the antecedent of an anaphoric definite. Additionally, the antecedent of an anaphoric definite must be compatible with the definite's function. For example, if the antecedent is universally quantified, the antecedent's variable must be contained in the definite function's argument list. Notice that this routine differs from the pronoun compatibility routine. Not all pronoun antecedents can be antecedents for anaphoric definites.

Armed with these two compatibility routines, our program screens out noun phrases that could never be antecedents for a pronoun or a definite noun phrase in the same sentence. The logical form for a pronoun or anaphoric definite is quite useful for filtering inappropriate antecedents. However, to provide a full blown theory of anaphora resolution, we must also indicate the effect of context and pragmatics on antecedent choice. This is an issue beyond the scope of this thesis.

6.3.2 Non-anaphoric Definites and Indefinites

In order to provide the final meaning of a trigger sentence, the program must limit the initial representations of non-anaphoric definites and indefinites. In this section, we describe the routines that convert the initial representations of definites and indefinites into their final meanings. First, we discuss how the program limits definite functions (as discussed in Chapter three).

To limit the meaning of an initial definite function, the program must decide whether the definite is used anaphorically or not. Before making this decision, the program must resolve the meanings

of all noun phrases embedded in the definite. The antecedents for all anaphoric noun phrases must be specified. The meanings of all embedded non-anaphoric definites must be pinpointed. We must also determine whether embedded quantified noun phrases (not subject to the complex noun phrase constraint) distribute over the definite. Finally, all embedded indefinites must be converted to functional form. Once this information is determined, we consider whether the restriction of the definite function contains an unbound variable. If it does, then that definite cannot be anaphoric and its function must be limited by argument reduction. To argument reduce a function, we determine the necessary arguments of the function. The necessary arguments for a definite function consist of all those quantified variables and function arguments (other than its own) not bound by quantifiers in the definite's restriction. Once the program determines the necessary arguments for a definite function, the argument reduction routine replaces the original composite function with a function over the necessary arguments. Hence, the program replaces the initial function corresponding to a non-anaphoric definite with a function whose argument list contains only the necessary arguments. While the initial functional representation of a definite is a composite, the final functional representation captures precisely the intended meaning of the non-anaphoric definite given the meanings of embedded noun phrases.

On the other hand, if the restriction of the definite function does not contain an unbound variable, the definite could be anaphoric, though not necessarily. Hence, when the restriction is closed, the program user determines whether the definite noun phrase is anaphoric or not. If the user decides that it is anaphoric, then the definite's function is equated with the discourse entity, variable, or function corresponding to its antecedent. The argument list of the anaphoric definite's function must be compatible with the representation of its antecedent even though its restriction is closed. On the other hand, if the user decides that a definite is non-anaphoric, the function is replaced with a function with no arguments (since the restriction contains no unbound variables).

Another important routine converts indefinites into functions (as discussed in Chapter four). Since the only quantified noun phrases we consider are universals and indefinites, once indefinites are converted into functions, quantifier scope ambiguity is specified. Also, by replacing indefinites with functions, the program provides another way to establish compatibility between an anaphoric noun phrase and an indefinite. Even if an indefinite is initially incompatible with a pronoun because its variable is not contained in the pronoun function's argument list, the indefinite's function may be compatible with the pronoun. This is an important part of handling donkey sentences.

We replace the variables of an existential operator with a function after determining those operators that have scope over the existential operator. Before doing so, we must determine the meanings for all noun phrases embedded in the indefinite. Once this is complete, any variables not bound by operators in the existential's restriction must be bound by operators outside of the existential. Hence, those variables should be included in the argument list of the indefinite function. Additionally, we must decide whether any universal operators or lambda operators that can have scope over the existential operator do have scope over it. Once these decisions are made, we provide a new representation of the indefinite as a function whose argument list consists of those variables corresponding to operators that have scope over it (either to bind a variable in the restriction or because the user decides the operator has scope over it). We substitute this function for all occurrences of the existential variable. Because the sentences we handle do not contain negation, the program does not check to see if an existential is in the scope of negation. In order to handle examples with negation, we should also consider whether any negation has scope over the existential before allowing the conversion.

6.3.3 Ellipsis Routines

To determine the meaning of an elided sentence, the program must locate its trigger sentence. Additionally, the meaning of the trigger sentence must be fully specified before we can provide the meaning of the elided sentence. Our program provides the meaning of the elided sentence by locating the trigger sentence, determining its meaning, and then replacing the elided verb phrase with the trigger verb phrase.

The program contains a routine to locate the intended trigger verb phrase (with user intervention), once the trigger sentence is disambiguated. When the trigger verb phrase is found, it is checked to ensure that the multiple pronoun constraint (discussed in Chapter five) holds and that all variables in the verb phrase are bound[2]. Additionally, we must ensure that the voice of the trigger verb phrase is compatible with the voice of the elided verb phrase[3]. The parser encodes the voice of all sentences converted to logical form. If a sentence has active voice, the surface subject fills the role of logical subject (expressed by adding the slot-filler pair (subject lambda-variable)). If the sentence has passive voice, it fills the role of logical object (expressed by adding the slot-filler pair (object lambda-variable)). This information allows the program to ensure that a passive voice verb phrase is not the trigger for an active voice elided verb phrase, or vice versa. The argument structures would not be compatible. Once all these checks are made, another routine replaces the elided verb phrase with the final meaning of the trigger verb phrase.

6.4 Example

In this section, we illustrate some features of our implementation by demonstrating how the program handles example 106.

Example 106

Fred$_i$ gave the psychiatrist who cares for (his$_i$ mother)$_j$ her$_j$ diary.
George$_k$ did too.
Meanings:
1. George gave the psychiatrist who cares for Fred's mother Fred's mother's diary.
2. George gave the psychiatrist who cares for George's mother George's mother's diary.
3. *George gave the psychiatrist who cares for George's mother Fred's mother's diary.
4. *George gave the psychiatrist who cares for Fred's mother George's mother's diary.

The program processes the sentences in this example, one at a time. First, it parses the trigger sentence. Once a parse tree is available, the parser outputs the logical form for the sentence.

[2]This program only deals with intersentential verb phrase ellipsis. Hence, all variables in the trigger verb phrase must be bound in the verb phrase, otherwise the elided sentence cannot receive a meaning. If we augment our approach to handle antecedent-contained ellipsis, we would have to allow unbound variables in the trigger verb phrase. However, these variables would have to be bound by an operator outside of the verb phrase but inside the meaning of the sentence.

[3]Voice compatibility is the only thing we check for in this implementation. Other factors are involved in determining whether a trigger verb phrase is compatible with an elided verb phrase. However, this was not a concern of this implementation.

Because the trigger sentence in example 106 has a single parse tree[4], the parser outputs a single logical form for the sentence, shown in 371.

Example 371

Fred gave the psychiatrist who cares for his mother her diary.

```
((def1) | (name (def1) Fred),
   λ(y)(give (subject y)
             (object ((def5 y) | (and (diary (def5 y))
                                      (possess (her6 y) (def5 y)))))
             (indirect-object
               ((def2 y) |
                  (and (psychiatrist (def2 y))
                       (def2 y),
                         λ(z)(care (subject z)
                                   (for ((def3 y z) |
                                           (and (mother (def3 y z))
                                                (possess (his4 y z)
                                                         (def3 y z))))))))))
```

Notice that the parser indicates the logical roles of noun phrases with slot-filler notation in the predicate-argument structure. Additional processing of the logical form in 371 is put off until the final meaning of the sentence is needed (i.e., when the meaning of a sentence with verb phrase ellipsis must be determined). Ideally, processing should be done as information becomes available. However, in this implementation, it is simpler to add additional information to the logical form of a trigger sentence only when its meaning must be determined to provide the meaning of an elided verb phrase. Hence, the logical form in 371 is saved on a stack of recently processed sentences.

Once the program saves the logical form of the trigger sentence on the stack, the parser processes the elided sentence and outputs a logical form for it. The fact that the representation contains an elided verb phrase is indicated in its representation, as can be seen in 372.

Example 372

```
((def7) | (name (def7) George)), λ(w)(dummy33 (subject w))
```

The word *dummy*$_{33}$ is the predicate in the predicate-argument structure for an elided verb phrase. To determine the meaning of the elided sentence in 372, the program searches through the stack of logical forms for the trigger sentence. The most recent sentences are the best candidates. Hence, the program presents the top-most logical forms, one at a time, to the user and asks him/her to decide whether it is the trigger sentence for the elided sentence in 372. Assume the user decides the logical form shown in 371 is the trigger sentence. Once the program locates the trigger sentence, it must disambiguate that trigger sentence before using it to provide the meaning for the elided verb phrase. We will attempt to convey the flavor of this disambiguation process in the next few paragraphs.

To decide on the final meaning of the logical form in 371, the program must obtain additional information. First, the program attempts to find antecedents for all of the pronoun functions (in

[4]If there is more than one parse tree, the program asks the user to pick the intended parse tree (and hence, the logical form).

this case, (her_6 y) and (his_4 y z)). Once the pronouns have antecedents, the program handles all definites, and then all quantified noun phrases. However, this control sequence is only realized when there is no interdependency between noun phrases in the sentence (e.g., the antecedents for all pronouns and anaphoric definites are outside of the current sentence). Once the program attempts to determine whether a pronoun can be anaphorically dependent on a definite or indefinite noun phrase in the same sentence, the flow of control can change. Often, additional information about a possible antecedent must be determined in order to decide if it is compatible with the function representing the anaphora.

The program must determine the meaning of the trigger sentence represented in 371. Suppose it begins by searching for the antecedent of the pronoun function (her_6 y). First, the program asks the user whether the antecedent occurs in a previous sentence. If so, the user also provides the discourse entity for that antecedent. Assume that the user decides that the antecedent is in the same sentence as the pronoun. Hence, the program examines each noun phrase in the logical form to determine whether any of them (i.e., (def_1), (def_5 y), (def_2 y), (his_4 y z), or (def_3 y z)) is compatible with the pronoun. The function (def_5 y) cannot be the antecedent for (her_6 y) because the pronoun is embedded in that function's restriction. Gender information could be used to rule out (def_1), and (his_4 y z) as antecedents. However, this information was not provided by the lexicon of the parser we used[5]. Assume that the user rules both of them out. The remaining antecedent candidates are (def_2 y) and (def_3 y z). Though (def_2 y) is a reasonable antecedent candidate, assume the user rules it out. The only remaining candidate is (def_3 y z), which is not immediately compatible with (her_6 y). However, because (def_3 y z) is a composite and has not been limited to its final meaning, there is still a chance that it could be compatible with the pronoun function after argument reduction or anaphora resolution. Hence, the program shifts its attention to (def_3 y z).

To further process (def_3 y z), the program must determine whether the restriction of that function contains any unbound variables. To determine the necessary arguments, the meaning of each embedded noun phrase must be pinpointed. Because the function contains a single embedded pronoun function (i.e., (his_4 y z)) with no assigned antecedent, the program must find that pronoun's antecedent before the behavior of the definite function is pinpointed. As before, the program asks the user if the antecedent for the pronoun occurs in a previous sentence. Assume a negative reply. Again, the program must determine which of the noun phrases in the sentence are compatible with the pronoun function. In this case, only (def_1), (def_5 y), (her_6 y), and (def_2 y) are possible antecedents. Assume that the user picks (def_1) as the antecedent. Since the pronoun function is compatible with (def_1), this information can be added to the logical form for the sentence. However, because (def_1) is a subject, the pronoun can refer to it directly or indirectly. The program asks the user to decide whether the pronoun reference is direct (leading to the replacement of the pronoun function by the definite function) or indirect (leading to the replacement of the pronoun function by the lambda variable). Assuming the user takes the indirect option, the logical form is updated as shown in example 373.

[5]Though this feature could easily be added to the lexicon.

Example 373

Fred$_i$ gave the psychiatrist who cares for his$_i$ mother her diary.

```
((def1) | (name (def1) Fred),
   λ(y)(give (subject y)
             (object ((def5 y) | (and (diary (def5 y))
                                      (possess (her6 y) (def5 y)))))
             (indirect-object
               ((def2 y) |
                  (and (psychiatrist (def2 y))
                       (def2 y),
                         λ(z)(care (subject z)
                                   (for ((def3 y z) |
                                           (and (mother (def3 y z))
                                                (possess  y (def3 y z))))))))))
```

Notice that the pronoun is replaced by its value in the logical form (as that logical form would be printed). However, the program keeps track of pronouns and their values for the multiple pronoun constraint.

Because the program has determined the meanings of all of the noun phrases embedded in (def$_3$ y z), it can finish processing that function. Since the restriction contains an unbound variable, that function cannot be anaphoric. However, given that the only necessary argument is *y*, the function (def$_3$ y z) is replaced by a function of *y*, as shown in 374.

Example 374

Fred$_i$ gave the psychiatrist who cares for his$_i$ mother her diary.

```
((def1) | (name (def1) Fred),
   λ(y)(give (subject y)
             (object ((def5 y) | (and (diary (def5 y))
                                      (possess (her6 y) (def5 y)))))
             (indirect-object
               ((def2 y) |
                  (and (psychiatrist (def2 y))
                       (def2 y),
                         λ(z)(care (subject z)
                                   (for ( (def7 y) |
                                            (and (mother (def7 y))
                                                 (possess y (def7 y))))))))))
```

Notice that the function (def$_3$ y z) is replaced by (def$_7$ y).

Because (def$_7$ y) replaces (def$_3$ y z) as the meaning for *his mother*, the program determines that *his mother* can be the antecedent for *her*. Assume the user confirms that *his mother* is the antecedent for *her*. Since *his mother*, now represented as (def$_7$ y), is not a subject, the program indicates that the antecedent for (her$_6$ y) is (def$_7$ y) in the logical form, as shown in 375.

Example 375

Fred$_i$ gave the psychiatrist who cares for (his$_i$ mother)$_j$ her$_j$ diary.

```
((def1) | (name (def1) Fred),
   λ(y)(give (subject y)
             (object ((def5 y) | (and (diary (def5 y))
                                      (possess (def7 y) (def5 y)))))
             (indirect-object
               ((def2 y) |
                  (and (psychiatrist (def2 y))
                       (def2 y),
                         λ(z)(care (subject z)
                                   (for ((def7 y) |
                                           (and (mother (def7 y))
                                                (possess y (def7 y))))))))))
```

Notice that (def$_7$ y) replaces (her$_6$ y) in the logical form.

To finish processing the sentence, the program must specify the meanings of the functions (def$_2$ y), (def$_5$ y), and (def$_1$). First, it must determine the necessary arguments (i.e., all of the unbound variables in the restriction) for each function. Notice that y is a necessary argument for both (def$_2$ y) and (def$_5$ y). Whenever a definite contains an unbound variable in its restriction, that definite cannot be anaphoric. Hence, (def$_2$ y) and (def$_5$ y) cannot be anaphoric. Additionally, since the argument lists of both functions contain only the necessary arguments, the functions do not need to be reduced. In contrast, the restriction of (def$_1$) does not contain any unbound variables, so it could be anaphoric. Hence, the program asks the user to determine whether that definite is anaphoric. Assume a negative answer. Since the function contains no arguments, there is no need to replace it with a new function.

Next, the program asks the user to determine which of the embedded verb phrases in the trigger sentence is the trigger for the elided verb phrase. In this case there is only one possibility, shown in 376.

Example 376

```
λ(y)(give (subject y)
          (object ((def5 y) | (and (diary (def5 y))
                                   (possess (def7 y) (def5 y)))))
          (indirect-object
            ((def2 y) |
               (and (psychiatrist (def2 y))
                    (def2 y),
                      λ(z)(care (subject z)
                                (for ((def7 y) |
                                        (and (mother (def7 y))
                                             (possess y (def7 y))))))))))
```

The program checks the verb phrase representation in 376 to ensure that it contains no unbound variables and that the multiple pronoun constraint holds. Notice the verb phrase in 376 contains no unbound variables and the multiple pronoun constraint does not apply since both pronoun functions have different antecedents. Finally, the program checks to ensure that the the trigger verb phrase in 376 is compatible in voice to the elided verb phrase in 372. Since the logical forms of

the trigger verb phrase in 376 and the elided verb phrase in 372 both indicate active voice[6], the program replaces the elided verb phrase with the trigger verb phrase, as shown in 377.

Example 377

```
((def7) | (name (def7) George)),
   λ(y)(give (subject y)
             (object ((def5 y) | (and (diary (def5 y))
                                      (possess (def7 y) (def5 y)))))
             (indirect-object
               ((def2 y) |
                  (and (psychiatrist (def2 y))
                       (def2 y),
                         λ(z)(care (subject z)
                                   (for ((def7 y) |
                                           (and (mother (def7 y))
                                                (possess y (def7 y))))))))))
```

Notice that the dummy verb phrase in 372 is replaced by the verb phrase in 376. Thus, we have provided the meaning of the elided sentence.

In this Chapter, we described how our approach is incorporated into a program that determines meanings of sentences with verb phrase ellipsis. In section 6.2, we described the routines necessary to generate the logical form for a sentence as it is parsed. Once the logical form is output by the parser, it is processed by the second part of the program, whose goal is to determine the meaning of sentences with verb phrase ellipsis. If a logical form contains a dummy predicate, the program locates its trigger sentence, disambiguates that trigger sentence, and then replaces the dummy verb phrase with the appropriate verb phrase from the trigger sentence. The routines that are necessary to disambiguate the trigger sentence were described in section 6.3. Finally, in section 6.4, we discussed how the system handles an example. We provided details of the flow of control, and information about how the compatibility routines and argument reduction routines work for that example. The program updates logical form on a need-to-know basis. If it needs to determine whether a certain noun phrase can be the antecedent for a pronoun, more information about the meaning of that noun phrase may be required to determine compatibility. Gathering information on a need-to-know basis is necessary because resolving anaphora, determining quantifier scoping, and determining the final meanings for a definite noun phrases are interdependent processes.

[6]Each logical form indicates that its sentence has active voice because the surface subject fills the role of logical subject. If a sentence has passive voice, its surface subject would fill the role of logical object.

Chapter 7

Conclusions and Future Directions

7.1 Summary

In this thesis, we explored the usefulness of logical form for developing a computer model of language. We introduced guidelines for properly exploiting logical form in a computer model. These guidelines constrained the representations we provided for pronouns, definites, and indefinites. The compactness constraint requires that we provide a single representation for an ambiguous sentence (and its parts). The modularity constraint requires that the representations for a sentence and its parts be derived using only syntactic and sentence-level information. Finally, the formal consistency constraint requires that any update of logical form should simply make the meaning of the sentence more precise. In addition to obeying these constraints, we were concerned with providing representations that accurately model the ways that noun phrases behave in English. Hence, our computational constraints and accurate modeling of language were primary concerns in this thesis.

In Chapter two, we discussed the representation of pronouns as functions in logical form. We demonstrated that this representation models the linguistic behavior of pronouns and that it satisfies our three computational constraints. The functional representation for a pronoun is a composite representation, compatible with any behavior it could have given its location in a sentence. We do not select a pronoun's representation depending on its antecedent. The function summarizes all possible behaviors simultaneously. Hence, the initial representation of a pronoun as a function satisfies our three constraints on logical form. Because we do not need to know the antecedent for the pronoun before providing the representation, the compactness and modularity constraints are obeyed. Also, in keeping with the modularity constraint, we use only syntactic and sentence-level information to provide the logical form for a pronoun. Later, when we have enough information to pick the antecedent for the pronoun, the initial composite function is equated with a value depending on the type of its antecedent. Because of the way we designed the initial representation of a pronoun, any time we replace the pronoun function with its value, we obey the formal consistency constraint.

It is not surprising that other pronoun representations fail to obey our logical form constraints. Previous pronoun representations were usually developed for linguistic theories. Our constraints are much more important for computer models of language. When building a computer model for language, it is useful to provide a representation for a sentence's meaning which reflects only what we know by examining that sentence. The final meaning of a sentence is very difficult to determine without looking at previous and later sentences. Thus, a pronoun must be represented in some way

that captures how that pronoun can act without committing to an antecedent. The antecedent could be in another sentence or the choice of antecedent could require information provided in a later sentence.

In Chapter two, we also discussed the idea that pronouns adopt the behaviors prescribed by their antecedents. For example, if the antecedent of a pronoun is a universal, then the pronoun acts like a variable bound by a universal quantifier. Additionally, if the antecedent is a definite noun phrase, the pronoun must adopt the behavior of its definite antecedent. Hence, we discussed the ways that pronouns act when their antecedents are definite. Many researchers have suggested that pronouns must either act as constants or variables. However, we examined examples that present problems for the idea that pronouns whose antecedents are definite noun phrases must either act like bound variables or constants. Because of these examples, we considered whether some aspect of definite noun phrase behavior was missing from other models. Our discussion of definite noun phrases not only relates to how pronouns with definite antecedents act, but because a pronoun's behavior depends on the representation of its antecedent, this discussion also affects our representation of definite noun phrases discussed in Chapter three.

In Chapter three, we discussed the representation of singular definites as functions. We demonstrated that this representation models the linguistic behavior of singular definites and that it satisfies our three computational constraints. The functional representation for a definite is a composite representation, compatible with any behavior the definite could have given its location in a sentence. Initially, definites are represented as functions of all of the variables that could possibly affect their final meaning. Because a definite can be anaphoric, it is not surprising that the initial representation of a definite should be similar to a pronoun. However, definite functions differ from pronoun functions. Because definite noun phrases can be structurally complex, they often contain noun phrases which affect their behavior. These embedded noun phrases must be considered when we provide a functional representation for a definite noun phrase. Later when more information is available about the definite's meaning, the composite function is replaced with some value which precisely captures its intended meaning. This choice depends on whether the definite is anaphoric or not. If the definite is anaphoric, the definite function is replaced by a value depending on the type of its antecedent. On the other hand, if the definite is not anaphoric, it is replaced by a function of the necessary arguments. The necessary arguments are precisely those variables that are unbound in the definite's restriction. This allows us to pinpoint the meaning of the initial definite function in a way that is consistent with the meaning of a definite description containing variables but without many of the problems associated with definite descriptions.

The initial representation of a definite is provided using only syntactic and sentence-level information in keeping with the modularity constraint. It is compatible with all of the behaviors of a definite without committing us to one in particular, in keeping with the compactness constraint. Finally, when the initial representation is limited to its final meaning, we obey the formal consistency constraint. Because our representation of a definite obeys these constraints, it is a very attractive representation for someone who wants to build the meaning of a definite in a sentence incrementally. To provide the initial representation for a definite, we do not have to decide whether it is anaphoric or not. Also, we do not have to know how embedded noun phrases will behave before representing the definite. Later, when we want to pinpoint the meaning of a definite, this information must be obtained. However, initially we provide a composite representation for the definite using only syntactic and sentence-level information.

In Chapter four, we discussed the representation of indefinites as existentially quantified and restricted variables. This representation was necessary, otherwise we could not handle a variety

of intrasentential indefinite behaviors. However, because indefinites share many properties with definites, we considered how best to update the initial representation in order to model those indefinite behaviors. We chose to replace all existentially quantified variables with a function whose argument list includes all of the variables of universal quantifiers and lambda operators that have scope over the existential operator. This replacement is meaning-preserving. Hence, we are able to update our initial representation for an indefinite once the meanings of embedded noun phrases are determined and scoping information is obtained. Additionally, this update allows pronouns to be anaphorically dependent on indefinites embedded in relative clauses. So long as the function replacing an existentially quantified variable is compatible with a pronoun's function, the indefinite can be the pronoun's antecedent.

Because we used Allen's [1] method to put off quantifier scoping decisions, the initial representation of an indefinite as an existentially quantified variable is compatible with the compactness constraint. The quantificational representation of an indefinite is provided using only syntactic information, in keeping with the modularity constraint. We do not need to know what has scope over the indefinite. We do not need to know how embedded noun phrases will behave. Later when more information is available about embedded noun phrases and about what has scope over the existential, we can replace the existential's variables with a function. This function is compatible in meaning with the existential. However, an indefinite function, because of its limited argument list, may be accessible as the antecedent for a pronoun even though its existential quantifier cannot bind the pronoun.

In Chapter five, we discussed the multiple pronoun constraint for verb phrase ellipsis. Though this constraint does not affect how we represent pronouns in logical form, it does help us to eliminate some rather odd meanings for an elided sentence when its trigger verb phrase contains two or more pronouns whose antecedents are the subject of the sentence. This constraint filters out representations of trigger verb phrases which cannot be used to generate reasonable meanings for elided verb phrases.

In Chapter six, we summarized a program that we wrote to generate logical forms for sentences and meanings for sentences with verb phrase ellipsis. This implementation uses the representations for noun phrases discussed in the previous chapters. These representations are easily generated when the parser provides the parse tree for a sentence. Once the logical form for a sentence is generated, it is examined for verb phrase ellipsis. If the sentence is elided, the program searches the stack of recently processed sentences for its trigger. When the trigger sentence is located, the meaning of each noun phrase is determined. Antecedents for anaphoric noun phrases are located. The meanings of non-anaphoric definites are specified, and indefinites are converted into a functional representation. Finally, the program determines the meaning of the elided sentence, using the appropriate verb phrase from the disambiguated trigger sentence's representation.

Our program illustrates how dynamic anaphora resolution is. Many antecedents for pronouns (or anaphoric definites) are initially compatible with the pronoun functions. However, some definite and indefinite noun phrases may not be immediately accessible[1] (e.g., Bach-Peters sentences or donkey sentences). Our initial representation of a definite noun phrase is a composite representation consistent with all of the behaviors that definite can have given its position in a sentence. Thus, a composite definite function could be initially incompatible with a certain pronoun function. However, later when we know more about the meanings of noun phrases embedded in that definite, we can pinpoint its meaning. The reduced definite function may be accessible to the pronoun

[1]It might be interesting to determine whether people can more quickly accept antecedents that are immediately accessible to a pronoun function than those that are not. A reaction time study could lend psychological validity to our approach.

function that the composite definite function is inaccessible to. An indefinite is initially represented as an existentially quantified variable (without scoping information indicated). The existential operator corresponding to the indefinite may not be able to bind a pronoun in the same sentence (i.e., the pronoun's function does not include the existential variable). However, once we determine what has scope over the existential operator, we can replace the existential operator's variables with a function. This function may be accessible to the pronoun function.

Our program demonstrates why it is difficult to make c-command handle the variety of anaphora examples we discuss in this thesis. C-command uses only syntactic information to determine when something is immediately accessible to a pronoun. Sometimes, we need to know antecedents for pronouns embedded in a noun phrase or what has scope over the noun phrase to decide whether it can be the antecedent for a certain pronoun (or an anaphoric definite). Anaphora resolution is a dynamic part of mapping a sentence to its final unambiguous meaning. Though syntax constrains what is possible. Additional information is often required to decide whether some noun phrases can be antecedents for anaphoric noun phrases in a sentence.

7.2 Future Directions

In this thesis, we specified the representations of pronouns and singular definite and indefinite noun phrases in logical form. One obvious next step would be to expand the model to handle plurals as well. Plural noun phrases have been investigated by many researchers in the linguistics and artificial intelligence communities (e.g., [41], [49]). However, no one has attempted to provide a logical form representation for plural noun phrases consistent with our computational constraints. Additionally, we considered only universal, indefinite, and definite noun phrases. There are many other types of *quantified* noun phrases in English (e.g., *most*, *many*, *few*, etc.). These noun phrases should be incorporated into our model. Finally, we have limited our attention to syntactic and sentence-level constraints on the meanings of noun phrases. It may well be that the meanings of other types of linguistic expressions are syntactically constrained. The question is, how much of the meaning of a sentence can be captured using only syntactic and sentence-level information? This question deserves investigation.

In this thesis, we specified how to update logical form to indicate its final meaning. However, we did not provide a model for determining the final meanings of noun phrases. To determine the final meaning of a noun phrase in a sentence, we must be able to determine antecedents for embedded pronouns, we must be able to decide whether an embedded quantified noun phrase (not subject to the complex noun phrase constraint) has scope over its container, and we must be able to decide what has scope over an indefinite noun phrase. In our implementation, we rely heavily on the program user's help for decisions requiring pragmatic and contextual information. The program can determine when a noun phrase is accessible to a pronoun or anaphoric definite. However, it cannot use contextual information to select a single antecedent for a pronoun. When more than one intrasentential antecedent is compatible with the pronoun, our program is unable to choose between them without help. Our program needs an anaphor resolution module which uses contextual information in conjunction with our pronoun and definite compatibility rules to competently select one single antecedent. We should investigate the possibility of improving our computer model by incorporating anaphora resolution techniques discussed in [25,26,48,6,7]. In addition to an anaphora resolution module, we must also provide a method which uses contextual information to determine quantifier scoping. If we can determine good heuristics for selecting antecedents and determining quantifier scoping, our program would be able to determine the final

meanings of noun phrases without user help.

Our verb phrase ellipsis model in this thesis is limited. For example, our model does not include a mechanism to select the trigger verb phrase for an elided sentence. Hence, we should investigate the factors that make a certain verb phrase a good trigger candidate for an elided verb phrase. Additionally, we only considered examples of intersentential ellipsis. It would be interesting to extend our model to handle antecedent-contained ellipsis. An example of antecedent-contained ellipsis is shown in 378.

Example 378

Mary$_i$ wanted her$_i$ daughter to read every book Louise did.

Notice that the trigger verb phrase occurs in the trigger sentence unlike the examples we discussed in this thesis. Finally, we would also like to handle pronouns of laziness, first noticed by Kartunnen [30]. The pronoun *it* in the following example is a lazy pronoun.

Example 379

Every married man$_i$ gives his$_i$ paycheck to his$_i$ wife.
Every bachelor$_j$ gives it to his$_j$ girlfriend.

Notice the pronoun *it* cannot mean *every married man's paycheck* even though the antecedent for the pronoun is *his paycheck*. We believe that sloppy pronouns are related to verb phrase ellipsis. Hence, the phenomenon should be able to be incorporated into our approach. This is certainly a problem for future investigation.

Appendix A

Examples Handled by Our Implementation

This appendix contains a list of examples handled by our program. First, consider the examples of verb phrase ellipsis we handled.

1. Fred$_i$ loves his$_j$ wife.
   George$_k$ does too.
   Meanings:
   j=i, direct: George loves Fred's wife.
   j=i indirect: George loves George's wife.
   j≠i, his$_j$=Entity24: George loves Entity24's wife.

2. Fred was hit by the apple.
   Bill did too.
   Meanings:
   Not possible.

3. Fred$_i$ thinks that he$_j$ is sick.
   Bill$_k$ does too.
   Meanings:
   j=i direct: Bill thinks that Fred is sick.
   j=i indirect: Bill thinks that Bill is sick.
   j≠i, he$_j$=Entity25: Bill thinks Entity25 is sick.

4. Fred$_i$ thinks that he$_j$ is sick.
   He$_j$ is.
   Meanings:
   j=i direct: Fred is sick.
   j=i indirect: Fred is sick.
   j≠i, he$_j$=Entity26: Entity26 is sick.

5. Mary$_i$ loves him$_j$.
   Every man$_k$ assumes that she$_i$ does.
   Meanings:
   j≠i, him$_j$=Entity27: Every man assumes that Mary loves Entity27.

6. Fred showed every girl$_i$ her$_j$ picture.
   Bill did too.
   Meanings:
   j=i: Bill showed every girl$_i$ her$_i$ picture.
   j≠i, her$_j$=Entity28: Bill showed every girl$_i$ Entity28's picture.

7. Fred$_i$ believes that Jack$_j$ loves him$_k$.
   Tom$_l$ does too.
   Meanings:
   k=i direct: Tom believes that Jack loves Fred.
   k=i indirect: Tom believes that Jack loves Tom.
   k=j: not possible
   k≠i, him$_k$=Entity29: Tom believes that Jack loves Entity29.

8. Fred$_i$ believes that Jack$_j$ loves him$_k$.
   But, Tom$_l$ does.
   Meanings:
   k=i direct: Tom loves Fred.
   k=i indirect: Tom loves Fred.
   k=j: not possible
   k≠i, him$_k$=Entity30: Tom loves Entity30.

9. Fred$_i$ loves himself$_j$.
   George$_k$ does too.
   Meanings:
   j=i direct: George loves Fred.
   j=i indirect: George loves George.
   j≠i: not possible

10. Fred$_i$ believes that Jack$_j$ loves himself$_k$.
    Tom$_l$ does too.
    Meanings:
    k=j direct: Tom believes that Jack loves Jack.
    k=j indirect: Tom believes that Jack loves Jack.
    k≠j: not possible

11. Fred$_i$ believes that Jack$_j$ loves himself$_k$.
    But, Tom$_l$ does.
    Meanings:
    k=j direct: Tom loves Jack.
    k=j indirect: Tom loves Tom.
    k≠j: not possible

12. Fred$_i$ persuaded every woman$_j$ that she$_k$ should go.
    Tom$_l$ did too.
    Meanings:
    k=j: Tom persuaded every woman$_j$ that she$_j$ should go.
    k≠j, she$_k$= Entity31: Tom persuaded every woman$_j$ that Entity31 should go.

13. Fred$_i$ persuaded every woman$_j$ that she$_k$ should go.
    So she$_l$ did.

Meanings:
k=j, l=j: not possible
$k \neq j$, l=k, she_k= Entity32: Entity32 went.

14. $Fred_i$ showed his_j $mother_k$ her_l picture.
$George_m$ did too.
Meanings:
j=i direct, l=k: George showed Fred's mother Fred's mother's picture.
j=i indirect, l=k: George showed George's mother George's mother's picture.
j=i direct, $l \neq k$, her_l=Entity33: George showed Fred's mother Entity33's picture.
j=i indirect, $l \neq k$, her_l=Entity33: George showed George's mother Entity33's picture.
his_i=Entity34, her_l=Entity35 George showed Entity34's mother Entity35's picture.

15. $Every\ man_i$ loves his_j mother.
$George_k$ does.
j=i direct: not possible
j=i indirect: George loves George's mother.
$j \neq i$, his_j=Entity36: George loves Entity36's mother.

16. Fred saw the girl.
George did too.
Meanings:
George saw the same girl as Fred saw.

17. George loves every man's wife.
Fred does too.
Meanings:
Fred loves the same wives as George loves.

18. $Fred_i$ believes that he_j is happy.
$George_k$ does too.
Meanings:
j=i direct: George believes that Fred is happy.
j=i indirect: George believes that George is happy.
$j \neq i$, he_j=Entity37: George believes that Entity37 is happy.

19. $Fred_i$ believes that he_j is happy.
And he_j is.
Meanings:
j=i direct: Fred is happy.
j=i indirect: Fred is happy.
$j \neq i$, he_j=Entity38: Entity38 is happy.

20. Fred loves the wife of a friend.
George does too.
Meanings:
George loves the same wife as Fred loves.
George loves the wife of a different friend.

21. Fred loves the woman who chased a cat.
George does too.

Meanings:
George loves the same woman as Fred loves.

22. Fred$_i$ loves the wife of every man he$_j$ knows.
George$_k$ does too.
Meanings:
j=i direct: George loves the wife of every man Fred knows.
j=i indirect: George loves the wife of every man George knows.
j≠i, he$_j$=Entity39: George loves the wife of every man Entity39 knows.

23. Fred told every girl$_j$ about her$_k$ mother.
George did too.
Meanings:
k=j: George told the same girls about their mothers.
k≠j, her$_k$=Entity40: George told the same group of girls about Entity40's mother.

24. Fred$_i$ showed the diary of (his$_j$ mother)$_k$ to (her$_l$ psychiatrist)$_m$.
Bill did too.
Meanings:
j=i direct, l=k: Bill showed the diary of Fred's mother to Fred's mother's psychiatrist.
j=i indirect, l=k: Bill showed the diary of Bill's mother to Bill's mother's psychiatrist.
j=i direct, l≠k, her$_l$= Entity41: Bill showed the diary of Fred's mother to Entity41's psychiatrist.
j=i indirect, l≠k, her$_l$= Entity41: Bill showed the diary of Bill's mother to Entity41's psychiatrist.
his$_j$=Entity43, her$_l$=Entity44: Bill showed the diary of Entity43's mother to Entity44's psychiatrist.

25. Fred$_i$ gave (the psychiatrist who$_l$ cares for (his$_j$ mother)$_k$)$_l$ her$_m$ diary.
George did too.
Meanings:
j=i direct, m=k: George gave the psychiatrist who cares for Fred's mother Fred's mother's diary.
j=i indirect, m=k: George gave the psychiatrist who cares for George's mother George's mother's diary.
j=i direct, m≠k, her$_m$=Entity45: George gave the psychiatrist who cares for Fred's mother Entity45's diary.
j=i indirect, m≠k, her$_m$=Entity45 : George gave the psychiatrist who cares for George's mother Entity45's diary.
j=l, m=k: George gave the psychiatrist who cares for the psychiatrist's mother the psychiatrist's mother's diary.
his$_j$=Entity46, m=k: George gave the psychiatrist who cares for Entity46's mother Entity46's mother's diary.
his$_j$=Entity46, her$_m$=Entity47: George gave the psychiatrist who cares for Entity46's mother Entity47's diary.

26. Fred saw a dog.
George did too.
Meanings:

George saw the same dog as Fred saw.
George saw a different dog than Fred saw.

27. Fred saw the dog.
George did too.
Meanings:
George saw the same dog as Fred saw.

28. Fred$_i$ saw a friend of his$_j$.
Tom did too.
Meanings:
j=i direct:
Tom saw the same friend of Fred's.
Tom saw a different friend of Fred's.
j=i indirect: Tom saw a friend of Tom's.
j≠i, his$_j$= Entity48:
Tom saw the same friend of Entity48's as Fred saw.
Tom saw a different friend of Entity48's than Fred saw.

29. Fred$_i$ saw his$_j$ friend.
Tom did too.
Meanings:
j=i direct: Tom saw Fred's friend.
j=i indirect: Tom saw Tom's friend.
j≠i, his$_j$= Entity49: Tom saw Entity49's friend.

30. Fred$_i$ persuaded Harry$_j$ to like him$_k$.
George$_l$ did too.
Meanings:
k=i direct: George persuaded Harry to like Fred.
k=i indirect: George persuaded Harry to like George.
k=j: not possible
him$_k$=Entity50: George persuaded Harry to like Entity50.

31. Fred$_i$ persuaded Harry$_j$ to like himself$_k$.
And he$_j$ does.
Meanings:
k=j direct: Harry likes Harry.
k=j indirect: Harry likes Harry.
k≠j: not possible

32. Fred$_i$ wanted to see himself$_j$.
George$_k$ did too.
Meanings:
j=i direct: George wanted to see Fred.
j=i indirect: George wanted to see George.
j≠i: not possible

33. Fred$_i$ wanted to see himself$_j$.
George$_k$ did.

Meanings:
j=i direct: George saw Fred.
j=i indirect: George saw George.
j≠i: not possible

34. A flag was hanging at every window.
A picture was too.
Meanings:
1. The same flag was hanging at every window.
The same picture was hanging at every window.
2. For every window a flag was hanging at it.
For every window a picture was hanging at it.

The following are examples we used to test the multiple pronoun constraint:

1. Fred$_i$ believed that he$_i$ loved his$_i$ wife.
Harry$_k$ did too.
Meanings:
Harry believed that Fred loved Fred's wife.
Harry believed that Harry loved Harry's wife.
Harry believed that Harry loved Fred's wife.

2. Fred$_i$ believed that his$_i$ wife loved him$_i$.
Harry$_k$ did too.
Meanings:
Harry believed that Fred's wife loved Fred.
Harry believed that Harry's wife loved Harry.
Harry believed that Fred's wife loved Harry.

3. Fred$_i$ said that he$_i$ believed that he$_i$ went.
George$_k$ did too.
Meanings:
George said that Fred believed that Fred went.
George said that George believed that George went.
George said that George believed that Fred went.

4. Fred$_i$ said that a friend of his$_i$ likes him$_i$.
Tom$_j$ did too.
Tom said that the same friend of Fred's likes Fred.
Tom said that a different friend of Fred's likes Fred.
Tom said that a friend of Tom's likes Tom.
Tom said that the same friend of Fred's likes Tom.
Tom said that a different friend of Fred's likes Tom.

5. Fred$_i$ said that every girl who he$_i$ knows likes him$_i$.
Tom$_j$ does too.
Tom said that every girl who Fred knows likes Fred.
Tom said that every girl who Tom knows likes Tom.
Tom said that every girl who Fred knows likes Tom.

6. Fred$_i$ told himself$_i$ about himself$_i$.
   George$_k$ did too.
   Meanings:
   George told Fred about Fred.
   George told George about George.

7. Fred$_i$ showed his$_i$ mother his$_i$ dog.
   George$_k$ did too.
   Meanings:
   George showed Fred's mother Fred's dog.
   George showed George's mother George's dog.
   George showed Fred's mother George's dog.
   George showed George's mother Fred's dog.

8. Fred$_i$ told every dog breeder$_j$ that the dog she$_j$ gave him$_i$ bit him$_i$.
   George$_k$ did too.
   Meanings:
   George told every dog breeder$_j$ that the dog she$_j$ gave Fred bit Fred.
   George told every dog breeder$_j$ that the dog she$_j$ gave George bit George.
   George told every dog breeder$_j$ that the dog she$_j$ gave Fred bit George.

These examples show the types of legal antecedents pronouns can have in our implementation (given pronoun resolution, gender information, and formal consistency):

1. Every boy$_i$ showed (his$_j$ mother)$_k$ her$_l$ clock.
   j can equal i or some index outside of the sentence.
   l can equal k or some index outside of the sentence.

2. (Every boy who$_i$ saw (his$_j$ cat)$_k$)$_i$ chased it$_l$.
   j can equal i or some index outside of the sentence.
   l can equal k or some index outside of the sentence.

3. (Every boy who$_i$ saw (his$_j$ cat)$_k$)$_i$ chased the cat$_l$.
   j can equal i or some index outside of the sentence.
   l can equal k or some index outside of the sentence.

4. Every boy$_i$ loves his$_j$ mother.
   j can equal i or some index outside of the sentence.

5. Every boy$_i$ loves the girl who loves him$_j$.
   j can equal i or some index outside of the sentence.

6. Every doctor$_i$ cares for the doctor$_j$.
   j can equal some index outside of the sentence.

7. Every man$_i$ gave (the psychiatrist who$_l$ cares for (his$_j$ mother)$_k$)$_l$ her$_m$ pill.
   j can equal i, l, or some index outside of the sentence.
   m can equal l, k, or some index outside of the sentence.

8. Every man$_i$ gave (the psychiatrist who$_k$ cares for every woman$_j$)$_k$ her$_l$ pill.
   l can equal k or some index outside of the sentence.

9. Every man$_i$ asked (his$_j$ mother)$_k$ about her$_l$ problem.
   j can equal i or some index outside of the sentence.
   l can equal k or some index outside of the sentence.

10. He$_i$ loves every man$_j$.
    i can equal an index outside of the sentence.

11. Every man$_i$ showed every boy$_j$ his$_k$ picture.
    k can equal i, j, or an index outside of the sentence.

12. Every man's$_i$ friend$_j$ loves him$_k$.
    k can equal i, but not j.

13. (The child who visits every man$_i$)$_j$ cares for him$_k$.
    k can equal some index outside of the sentence.

14. Every man$_i$ saw (the woman who showed every boy$_j$ his$_k$ dog)$_l$.
    k can equal i, j, or some index outside of the sentence.

15. Every man$_i$ told ((his$_j$ mother's)$_k$ psychiatrist)$_l$ about the old lady's$_m$ diary.
    j can equal i or some index outside of the sentence.
    m can equal k or some index outside of the sentence.

16. (The owner of every dog$_i$)$_j$ is afraid of the animal$_k$.
    k can equal i or some index outside of the sentence.

17. Every man$_i$ showed (his$_j$ mother's)$_k$ diary to (her$_l$ psychiatrist)$_m$.
    j can equal i or some index outside of the sentence.
    l can equal k or some index outside of the sentence.

18. Every man$_i$ gave (the psychiatrist who$_l$ cares for (his$_j$ mother)$_k$)$_l$ her$_m$ diary.
    j can equal i or l or some index outside of the sentence.
    m can equal l or k or some index outside of the sentence.

19. (Every miner who$_i$ loves (his$_j$ wife)$_k$)$_i$ cherishes her$_l$.
    j can equal i or some index outside of the sentence.
    l can equal k or some index outside of the sentence.

20. (Every man who$_i$ loves (his$_j$ mother)$_k$)$_i$ cares for the old lady$_l$.
    j can equal i or some index outside of the sentence.
    l can equal k or some index outside of the sentence.

21. (The boy who saw her$_j$)$_i$ kissed (the girl who loved him$_l$)$_k$.
    l can equal i or some index outside of the sentence.
    j can equal k or some index outside of the sentence.

22. Every man$_i$ told (her$_j$ mother)$_k$ that (his$_l$ wife)$_m$ should get a job.
    l can equal i or some index outside of the sentence.
    j can equal m or some index outside of the sentence.

23. (Her$_i$ mother)$_j$ told every man$_k$ that (his$_l$ wife)$_m$ should get a job.
    i can equal m (but only if l does not equal k) or it can equal some index outside the sentence.
    l can equal k or some index outside of the sentence.

24. (Every miner who owns a donkey$_j$)$_i$ beats it$_k$.
    k can equal j or some index outside of the sentence.

25. (Every miner who owns a donkey$_j$)$_i$ beats the animal$_k$.
    k can equal j or some index outside of the sentence.

The following examples show the variety sentences with quantifiers we handle:

1. George saw every student's paper.
2. George respects the author of every novel.
3. George respects the mother who cares for every boy.
4. Every man loves the child of a friend.
5. Every man chased the cat which chased a dog.
6. Some man loves every woman.
7. Every man wants some woman to be elected.
8. Every man believed that some woman was elected.
9. Some man wants every woman to be elected.
10. Some man believed that every woman was elected.
11. Every man believes that the girl who kicked a man was punished.
12. Every man believes that the child of a friend was punished.
13. Every man saw the boy with his binoculars.

Appendix B

Syntax and Semantics for Our Logical Form

In this appendix, we extend the syntax and semantics of first order logic (as given by Morgenstern [36]). We lambda-abstract syntactic subjects for the purpose of handling the sloppy identity ambiguity in verb phrase ellipsis (as discussed in Chapters two, three, and four). In other words, we represent a sentence S using the formula τ, $\lambda(\mathrm{w})\psi$, where τ is a term representing the subject and $\lambda(\mathrm{w})\psi$ is the lambda function representing the verb phrase. However, the meaning of the formula τ, $\lambda(\mathrm{w})\psi$ is simply $\psi(\tau/\mathrm{w})$, that is, ψ with each free occurrence of w replaced with τ. Before providing the truth value for a sentence, we must specify antecedents for anaphoric noun phrases (as discussed in Chapters two, three, and six) and specify quantifier scoping (as discussed in Chapter four). We must also eliminate the abstraction operator by applying the subject to the verb phrase. Before eliminating the lambda operator, we must determine whether it has scope over any existential operator in its scope (as discussed in Chapter four). The syntax and semantics of logical form are quite straight forward, with the exception of the fact that we use some syntactic sugar and place quantifiers inside the predicate argument structure. We will indicate the meanings of these forms in the semantics section.

B.1 Syntax

1. The logical constants: $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$,), (, =, $\forall$, $\exists$, :, |. We often use English equivalents of the logical constants (e.g., *and* for $\wedge$).

2. Non-logical constants: These include numerical constants (e.g., 1, 2, etc.), character constants (e.g., A, b, etc.), non-numerical, non-character constants (e.g., Fred34).

3. Variables: For example, x, y, z.

4. Predicate symbols: For example, run, boy, etc.

5. Function symbols: For example, def_4, indef_{66}, his_{66}.

We will also characterize terms, q-terms, atomic formulas, well-formed formulas, and sentences.

1. Terms: A term is any expression that refers to an object. Formally, they are defined by the following rules:

   (a) If τ is a constant, τ is a term.

   (b) If τ is a variable, τ is a term.

   (c) If τ_1, τ_2, ..., τ_n are terms, and ϕ is an n-ary function symbol, then $(\phi\ \tau_1\ \tau_2\ \ldots\ \tau_n)$ is a term. We allow a function to have a restriction (which is a well-formed formula). This is indicated by creating a list consisting of the function, |, and the restriction. For example, $((\phi\ \tau_1\ \tau_2\ \ldots\ \tau_n)\ |\ \psi))$

2. Q-term: We introduce the idea of a q-term to the syntax. A q-term is a quantifier, restriction (which is a well-formed formula), and variable used as a pseudo-term in a formula. The meaning of a q-term will be introduced in the section on semantics. For example, $[\forall\alpha\colon\ \psi\ \alpha]$ is a q-term, where α is a variable and ψ is a well-formed formula.

3. Atomic formula: If π is an n-ary predicate symbol, and τ_1, τ_2, ..., τ_n are terms or q-terms, then $(\pi\ \tau_1\ \tau_2\ \ldots\ \tau_n)$ is an atomic formula.

4. Well-formed formula: Well-formed formula are defined by the following formation rules:

   (a) If ϕ is an atomic formula, then it is a well-formed formula.

   (b) If ϕ is a well-formed formula $\tau_1{=}\tau_2$ and τ_1 and τ_2 are terms, then ϕ is a well-formed formula.

   (c) If ϕ is a well-formed formula, then $\neg\phi$ is a well-formed formula.

   (d) If ϕ and ψ are well-formed formulas, then (or $\phi\ \psi$), (and $\phi\ \psi$), (if $\phi\ \psi$), and (iff $\phi\ \psi$) are well-formed formulas.

   (e) If ϕ is a well-formed formula and α is a variable, then $\forall\alpha\ \phi$ and $\exists\alpha\ \phi$ are well-formed formula.

   (f) If ϕ is a formula and $[\forall\alpha\colon\ \psi\ \alpha]$ or $[\exists\alpha\colon\ \psi\ \alpha]$ is a q-term in the formula, then that formula is well-formed.

   (g) If ϕ is a well-formed formula and τ is a term (corresponding to the syntactic subject of the represented sentence) in the formula, then τ, $\lambda(x)\phi'$ (where ϕ' is $\phi(x/\tau)$) is a well-formed formula.

5. Sentences: Sentences are well-formed formulas that do not contain free variables. We must add that whenever we include a quantifier and its variable as a q-term in the predicate, then that quantifier is able to bind those variables inside the formula containing it.

B.2 Semantics

The model M for language L.

1. Domain D of objects in the world.

2. A mapping assigning each non-logical constant of the language a member of the domain.

3. A mapping assigning each n-ary predicate of L a set consisting of n-tuples that can be formed out of elements of D.

4. A mapping assigning each n-ary function of L a set of n+1 tuples formed from the elements of D.

We define the value of a constant term τ under interpretation M as follows:

1. If τ is a constant, then the value of τ under M is the element of D which M maps to τ.

2. If τ is of the form $\theta(\tau_1,\tau_2, ..., \tau_n)$ where θ is an n-ary function symbol, σ_1 is the value for τ_1, σ_2 is the value for τ_2,..., σ_n is the value for τ_n, and $[\sigma_1, \sigma_2, ..., \sigma_n, \sigma]$ is an element in the set of n+1 tuples that M maps to θ, then σ is the value of τ.

To this we add:

1. If $(\text{pro}_i\ \tau_1\ \tau_2\ ...\ \tau_n)$ occurs in a formula and τ_1, ..., τ_n are terms and some τ_i is a variable bound by $\lambda(\tau_i)$, the lambda operator has scope over the function. Also for any variable τ_j not bound by a lambda operator, then if there is an operator over τ_j contained as a q-term in the formula containing the pronoun function (or in a higher formula), then that operator op_j has scope over the pronoun function.

2. If $(\text{def}_i\ \tau_1\ \tau_2\ ...\ \tau_n)$ occurs in a formula ψ and τ_1, ..., τ_n are terms and τ_i is a variable bound by $\lambda(\tau_i)$, then the lambda operator has scope over the function. Also for any variable τ_j not bound by a lambda operator, then if there is an operator over τ_j contained in a q-term in the same formula (or in a higher formula) as the function but outside of the function's restriction, then that operator op_j has scope over the function. On the other hand, if τ_j is a variable whose operator is a universal or existential contained in the function's restriction, the function receives the following meaning:
$(\text{or } [\text{op}_j\ (\text{def}_i\ \tau_1\ \tau_2\ ...\ \tau_j\ ...\ \tau_n)]\ (\text{def}_i\ \tau_1\ \tau_2\ ...\ \tau_{j-1}\ \tau_{j+1}\ ...\ \tau_n))$

What it means for a sentence ϕ to be true under an interpretation M.

1. If ϕ is an atomic sentence (i.e., ϕ is of the form $(\pi\ \tau_1, ..., \tau_n)$ where σ_1 is the value of τ_1,..., and σ_n is the value of τ_n), then $M \models \phi$ if and only if $[\sigma_1, ..., \sigma_n]$ is a member of the set which M assigns to π.

2. If ϕ has the form $\neg\omega$, where ω is an atomic sentence of the form $\pi(\tau_1, ..., \tau_n)$ and σ_1 is the value of τ_1,..., and σ_n is the value of τ_n), then $M \models \phi$ if and only if $[\sigma_1, ..., \sigma_n]$ is in the antiextension of the set which M assigns to π.

3. If ϕ has the form $\psi \vee \chi$, $M \models \phi$ if and only if $M \models \psi$ or $M \models \chi$ or both.

4. If ϕ has the form $\psi \wedge \chi$, $M \models \phi$ if and only if $M \models \psi$ and $M \models \chi$.

5. If ϕ has the form $\psi \rightarrow \chi$, $M \models \phi$ if and only if $M \models \neg\psi$ or $M \models \chi$ or both.

6. If ϕ has the form $\psi \leftrightarrow \chi$, $M \models \phi$ if and only if $M \models \psi$ and $M \models \chi$ or $M \models \neg\psi$ and $M \models \neg\chi$.

7. For sentences of the form $\forall\alpha\psi$ or $\exists\alpha\psi$, we use β-variants. If M and M' are interpretations with identical domains, and β is a constant, M is a β-variant of M' if M and M' differ only in what they assign to β.
   1) If ϕ has the form $\forall\alpha\psi$, $M \models \phi$ if and only if, for all M', if M' is a β-variant of M, M' $\models$

$\psi(\beta/\alpha)$, where $\psi(\beta/\alpha)$ is the expression obtained by substituting β for all free occurrences α in ψ.
2) If ϕ has the form $\exists\alpha\psi$, M $\models \phi$ if and only if, for some M', if M' is a β-variant of M, M' $\models \psi(\beta/\alpha)$, where $\psi(\beta/\alpha)$ is the expression obtained by substituting β for all free occurrences α in ψ.

8. If ϕ is a formula τ, $\lambda(x)\psi$, then the M $\models \phi$ if and only if M $\models \psi(\tau/x)$.

Additionally:

1. If $((\phi\ \tau_1\ \tau_2\ \ldots\ \tau_n) \mid \psi_1)$ is contained in a formula ψ_2, the formula is equivalent to (and ψ_3 ψ_1), where ψ_1 is removed from ψ_2 to give ψ_3.

2. If $(\pi\ [op_1\alpha_1\colon \psi]\ \ldots)$ is a formula, then this is equivalent to:
   op_1 (and ψ $(\pi\ \alpha_1\ \ldots)$) if op_1 is $\exists$
   op_1 (if ψ $(\pi\ \alpha_1\ \ldots)$) if op_1 is $\forall$.

3. If $(\pi\ [op_1\alpha_1\ \alpha_1]\ [op_2\alpha_2\ \alpha_2]\ \ldots\ [op_n\alpha_n\ \alpha_n])$ is a formula, it means:
   (or $op_1\alpha_1\ op_2\alpha_2\ \ldots\ op_n\alpha_n\ (\pi\ \alpha_1\ \alpha_2\ \ldots\ \alpha_n)$
   $op_2\alpha_2\ op_1\alpha_1\ \ldots\ op_n\alpha_n\ (\pi\ \alpha_1\ \alpha_2\ \ldots\ \alpha_n)$
   .
   .
   .
   $op_2\alpha_2\ \ldots\ op_n\alpha_n\ op_1\alpha_1\ (\pi\ \alpha_1\ \alpha_2\ \ldots\ \alpha_n)$
   .
   .
   .
   $op_n\alpha_n\ \ldots\ op_2\alpha_2\ op_1\alpha_1\ (\pi\ \alpha_1\ \alpha_2\ \ldots\ \alpha_n))$

4. If $(\pi\ \ldots\ [op_j\alpha_j\colon \psi_j\ \alpha_j]\ \ldots\ [op_i\alpha_i\colon \psi_i\ \alpha_i]\ \ldots)$ and α_j is unbound in ψ_i then $op_j\alpha_j$ must have scope over op_i.

5. If τ_1, $\lambda(\alpha_1)(\pi\ \ldots\ [op_i\alpha_i\colon \psi_i\ \alpha_i]\ \ldots)$ and α_1 is unbound in ψ_i then $\lambda(\alpha_1)$ must have scope over op_i.

6. If τ_1, $\lambda(\alpha_1)(\pi\ \ldots\ [op_i\alpha_i\colon \psi_i\ \alpha_i]\ \ldots)$ and op_i is an existential operator, then either replace every α_i with a function of α_1 and every variable corresponding with an operator that has scope over it or replace every α_i with a function of those variables corresponding with operators that have scope over it excluding α_1.

Bibliography

[1] J. Allen. *Natural Language Understanding.* The Benjamin/Cummings Publishing Company, Menlo Park, CA, 1987.

[2] E. Bach. Problominalization. *Linguistic Inquiry*, 1:121–122, 1970.

[3] E. Bach and B. Partee. Anaphora and semantic structure. In J. Kreiman and A. E. Ojeda, editors, *Papers from the Parasession on Pronouns and Anaphora.* Chicago Linguistic Society, Chicago IL, 1980.

[4] E. Charniak. Toward a model of children's story comprehension. AI Laboratory TR-266, MIT, 1972.

[5] E. Charniak. Jack and Janet in search of a theory of knowledge. In *The Proceedings of the 3rd Joint Conference on Artificial Intelligence*, 1973.

[6] E. Charniak. A neat theory of marker passing. In *The Proceedings of the 5th national conference on artificial intelligence*, 1986.

[7] E. Charniak and R. Goldman. A logic for semantic interpretation. In *The Proceedings of the 26th annual meeting of the association for computational linguistics*, 1986.

[8] E. Charniak and D. McDermott. *Introduction to Artificial Intelligence.* Addison-Wesley, Reading, MA, 1985.

[9] N. Chomsky. *Reflections on Language.* Pantheon, New York, 1975.

[10] N. Chomsky. *Essays on Form and Interpretation.* North Holland, Amsterdam, 1977.

[11] N. Chomsky. *Lectures on Government and Binding.* Foris, Dordrecht, 1981.

[12] A. Church. *The Calculi of Lambda-Conversion.* Princeton University Press, Princeton, 1941.

[13] R. H. Cooper. The interpretation of pronouns. In F. Heny and H.S. Schnelle, editors, *Syntax and Semantics*, volume 10. Academic Press, New York, 1979.

[14] A. Cormack. VIP anaphora: Variables and scope. In F. Landman and F. Veltman, editors, *Varieties of Formal Semantics, Proceedings of the fourth Amsterdam Colloquium in September, 1982*, Dordrecht, Holland, 1984. Forris Publications.

[15] Ö. Dahl. How to open a sentence: Abstraction in natural language. *Logical Grammar Reports*, 12, 1974.

[16] D. R. Dowty, R. E. Wall, and S. Peters. *Introduction to Montague Semantics.* D. Reidel Publishing Company, Boston, MA, 1981.

[17] P. T. Geach. *Reference and Generality.* Cornell University Press, Ithaca, 1962.

[18] I. Heim. *The Semantics of Definite and Indefinite Noun Phrases.* PhD thesis, University of Massachusetts, 1982.

[19] J. Higgenbotham. Pronouns and bound variables. *Linguistic Inquiry*, 11:679–708, 1980.

[20] J. Hintikka. Quantifiers in logic and quantifiers in natural language. In E. Saarinen, editor, *Game Theoretical Semantics.* D. Reidel, Dordrecht, 1979.

[21] J. Hintikka. Quantifiers vs quantification theory. In E. Saarinen, editor, *Game Theoretical Semantics.* D. Reidel, Dordrecht, 1979.

[22] J. Hintikka. *Anaphora and Definite Descriptions: Two applications of Game-Theoretical semantics.* D. Reidel Publishing Company, Boston, 1985.

[23] P. Hirschbühler. VP deletion and across-the-board quantifier scope. *Papers from the North Eastern Linguistic Society*, 12:132–139, 1982.

[24] G. Hirst. *Anaphora in Natural Language Understanding: A Survey*, volume 119 of *Lecture Notes in Computer Science.* Springer-Verlag, New York, 1981.

[25] J. R. Hobbs. Pronoun resolution. Computer Science Department 76-1, City University of New York, 1976.

[26] J. R. Hobbs. Resolving pronoun references. *Lingua*, 44:311–338, 1978.

[27] N. Hornstein. *Logic as Grammar: An Approach to Meaning in Natural Language.* MIT Press, Cambridge, MA, 1984.

[28] P. Jacobson. *The Syntax of Crossing Coreference Sentences.* PhD thesis, University of California, Berkeley, 1979.

[29] H. Kamp. A theory of truth and semantic representation. In J. Groenendijk, T. Janssen, and M. Stokhof, editors, *Formal Methods in the Study of Language*, volume 1. Mathematische Centrum, Amsterdam, 1981.

[30] L. Kartunnen. Pronouns and variables. *CLS*, 5, 1969.

[31] E. Keenan. The functional principle: Generalizing the notion of subject-of. *CLS*, 10:298–309, 1974.

[32] E. Klein. VP ellipsis in DR theory. In J. Groenendijk, D. de Jongh, and M. Stokhof, editors, *Studies in Discourse Representation and the Theory of Generalized Quantifiers.* Foris, Dordrecht, 1987.

[33] D. Lewis. Score-keeping in a language game. In R. Bäuerle, U. Egli, and A. von Stechow, editors, *Semantics from different points of views.* Springer, Berlin, 1979.

[34] R. May. *Logical Form: Its Structure and Derivation.* MIT Press, Cambridge, MA, 1985.

[35] R. Montague. Universal grammar. *Theoria*, 36:373–398, 1970.

[36] L. Morgenstern. *Foundations of a Logic of Knowledge, Action, and Communication.* PhD thesis, New York University, 1988.

[37] B. L. Nash-Webber and R. Reiter. Anaphora and logical form: On formal meaning representations for natural language. In *The Proceedings of the 5th IJCAI*, pages 121–131, August 1977.

[38] B. Partee and E. Bach. Quantification, pronouns, and VP anaphora. In J. Groenendijk, T. Janssen, and M. Stokhof, editors, *Formal Methods in the Study of Language*, volume 1. Mathematische Centrum, Amsterdam, 1981.

[39] P. M. Postal. *Crossover Phenomena.* Holt, Reinhart, and Winston, New York, 1971.

[40] T. Reinhart. *Anaphora and Semantic Interpretation.* Croom Helm, London, 1983.

[41] C. Roberts. *Modal Subordination, Anaphora, and Distributivity.* PhD thesis, University of Massachusetts, 1987.

[42] J. R. Ross. *Constraints on Variables in Syntax.* PhD thesis, MIT, 1967.

[43] B. Russell. Reference. In J. F. Rosenberg and C. Travis, editors, *Readings in the Philosophy of Language.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.

[44] I. A. Sag. *Deletion and Logical Form.* PhD thesis, MIT, 1976.

[45] I. A. Sag. Partial variable assignment functions, verb phrase ellipsis, and the dispensability of logical form. 1981.

[46] L. K. Schubert and F. J. Pelletier. From English to logic : Context-free computation of 'conventional' logical translations. *American Journal of Computational Linguistics*, 10:165–176, 1984.

[47] P. Sells, A. Zaenen, and D. Zec. Reflexivization variation: Relations between syntax, semantics, and lexical structure. In M. Iida, S. Wechsler, and D. Zec, editors, *Studies in Grammatical Theory and Discourse Structure*, volume 1. CSLI, Stanford, to appear.

[48] C. Sidner. Focusing in the comprehension of definite anaphora. In M. Brady and R. Berwick, editors, *Computational Models in Discourse.* MIT Press, Cambridge, MA, 1983.

[49] B. L. Webber. *A Formal Approach to Discourse Anaphora.* PhD thesis, Harvard, 1978.

[50] B. L. Webber. So what can we talk about now? In M. Brady and R. Berwick, editors, *Computational Models of Discourse.* MIT Press, Cambridge MA, 1983.

[51] E. S. Williams. Discourse and logical form. *Linguistic Inquiry*, 8:101–139, 1977.