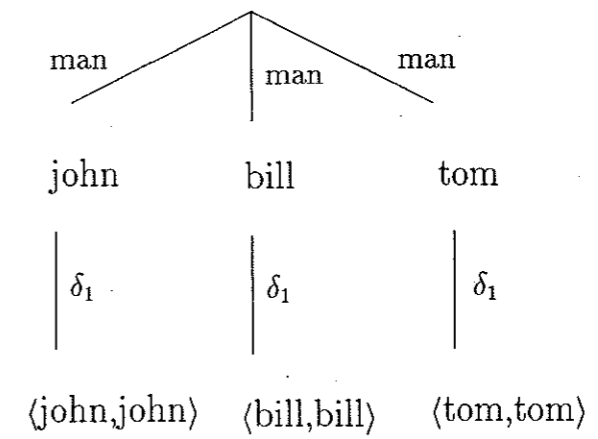


Semantic Trees



Dorit Ben-Shalom

UNIVERSITY OF CALIFORNIA

Los Angeles

Semantic Trees

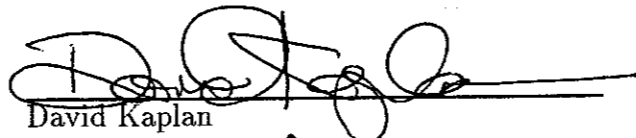
A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Linguistics

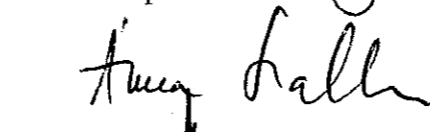
by


Dorit Ben-Shalom

1996

The dissertation of Dorit Ben-Shalom is approved.


David Kaplan


Anna Szabolcsi


Edward Keenan, Committee Co-chair


Edward Stabler, Committee Co-chair

University of California, Los Angeles

1996

Contents

1 Semantic Trees	1
1.1 Introduction	1
1.2 Semantic trees	2
1.3 A modal language	4
1.4 A fragment of English	5
1.5 Tree models	7
1.6 Pronouns and delta predicates	9
1.7 A road map	13
Appendix	15
References	18
2 Dependent and independent pronouns	19
2.1 Introduction	19
2.2 Independent vs. dependent pronouns	20
2.3 Coreference vs. binding	20
2.4 α - vs. β -occurrences	22
2.5 Names vs. delta predicates	24
2.6 Dependent α - and independent β -occurrences	26
2.7 Minimality	28

2.8 Conclusion	31
References	31
3 Static and dynamic binding	32
3.1 Introduction	32
3.2 Semantic trees	33
3.3 Predicate logic with anaphora	35
3.4 A combined system	37
3.5 Static and dynamic binding	38
3.6 Conclusion	40
References	40
4 Generalized quantifier reducibility	41
4.1 Introduction	41
4.2 Definitions	43
4.3 Graphic invariance	45
4.4 Some unreducible GQs	51
Appendix	65
References	66
5 Conservativity and extension	67
5.1 Introduction	67
5.2 Generalized quantifiers and modal operators	68
5.3 Conservativity and extension	68
5.4 Conclusion	69
References	70

ACKNOWLEDGMENTS

I would like to thank the members of my committee for their encouragement and criticisms. In particular, I would like to thank Ed Keenan for introducing me to generalized quantifiers, Anna Szabolcsi for a lot of what I know about natural language semantics, and Ed Stabler for teaching me to think abstractly about grammars and for serving as a role model in many ways. I would like to thank the people at CWI and FWI in Amsterdam for their hospitality and for introducing me to modal logic. I would like to thank all my friends at UCLA for being so nice.

Chapter 4 is re-formatted from my paper 'A tree characterization of generalized quantifier reducibility', in the CSLI (1994) volume *Polarity and Quantification* edited by Makoto Kanazawa and Christopher J. Piñón. Used with permission.

VITA

August 30, 1966	Born, Rehovot, Israel
1985–1987	Army Service, Israel
1990	M.A., Linguistics Tel Aviv University Tel Aviv, Israel
1991–1993	Teaching Assistant Department of Linguistics University of California, Los Angeles
1994	Guest Student Center for Mathematics and Computer Science Amsterdam
1994–1995	Dissertation Year Fellowship University of California, Los Angeles

PUBLICATIONS

- Beghelli, F., Ben-Shalom, D., and Szabolcsi, A. 1993. When do subjects and objects exhibit a branching reading?. *Proceedings of WCCFL XII*.
- Ben-Shalom, D. 1993. Object wide scope and semantic trees. *Proceedings of SALT III*.
- Ben-Shalom, D. 1994. Natural language, generalized quantifiers, and modal logic. *Proceedings of the 9th Amsterdam Colloquium*.

ABSTRACT OF THE DISSERTATION

Semantic Trees

by

Dorit Ben-Shalom

Doctor of Philosophy in Linguistics

University of California, Los Angeles, 1996

Professor Edward Keenan, Co-chair

Professor Edward Stabler, Co-chair

Let *logical form* be a neutral term for a syntactic level of linguistic representation which serves as the input to semantics. I assume that the grammar of natural language specifies the connections between the collection of logical form representations and three other types of objects: natural language tokens, models, and proofs.

A standard language for specifying logical form representations is predicate logic. In terms of the picture of the grammar above, predicate logic has a worked out proof theory but has problems in its connections with both natural language syntax and its own models. For example, it cannot capture the fact that the common noun *man* and the intransitive verb *walks* play different syntactic roles in the sentence *Every man walks*. In terms of its own semantics, the standard truth definition for predicate logic is stated indirectly with respect to assignments rather than directly with respect to models.

I introduce a variant of predicate logic that has a new language and a new class of models. The language is a propositional modal language; the models can be thought of as trees. The fact that the language is propositional and modal allows for a much closer fit with current Chomskian syntactic representations. The tree models allow for a direct, concrete semantics that does not involve the notion of an assignment.

In the overview chapter I use the new language to specify logical form representations for a simple fragment of English. To show that the perspective is of real linguistic interest, there are four independent applications of the perspective to current issues in natural language semantics. Two applications are about pronominal anaphora: Chapter 2 is about dependent and independent pronouns and Chapter 3 is about static and dynamic binding. The other two are about linguistic applications of generalized quantifier theory: Chapter 4 is about reducibility to iterations and Chapter 5 is about conservativity and extension.

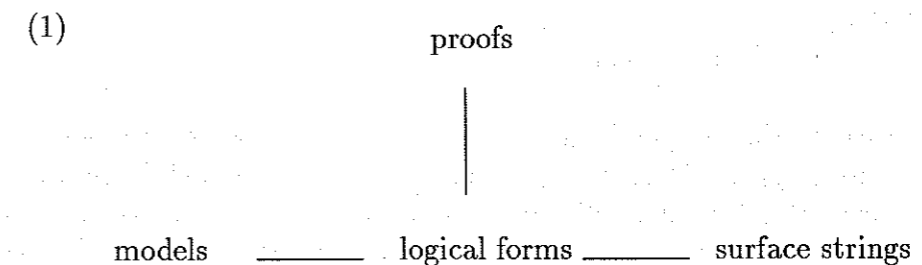
For this simple fragment, the connections between the four types of objects related by the grammar are simple: the logical form representations are close in form to standard syntactic representations; they have a standard semantics with respect to a class of intuitive models; and they are close enough to predicate logic formulas to make use of the latter's proof theory. Future research may tell how much of this simplicity can be maintained for larger fragments.

Chapter 1

Semantic Trees*

1.1 Introduction

Chomskian linguistics and Montague grammar are different theories about the grammar of natural language. They postulate radically different syntactic representations and do not even agree about the proper role of semantics within the grammar. But both of them use a syntactic level of representation which serves as the input to semantics. I call this level *logical form* and intend it as a neutral term for this level of linguistic representation. Following Stabler (to appear), I also assume that it is logical form that serves as the input to proof theory, i.e., to reasoning that is carried out by manipulating representations derived from natural language expressions. I thus assume that the relevant part of the grammar of natural language specifies the connections between the four types of objects in (1):



A standard language for specifying logical form representations is predicate logic. In terms of the grammar in (1), predicate logic has a worked out proof theory but has well-known problems in its connections with both natural language syntax and its own models. For example, predicate logic cannot capture the fact that the common noun *man* and the intransitive verb *walks* play different syntactic roles in the sentence *Every man walks*. In terms of its own semantics, the standard truth definition for predicate logic is stated indirectly with respect to assignments rather than directly with respect to models. I solve

*I would like to thank Nissim Francez, David Kaplan, Ed Keenan, Ed Stabler, and Anna Szabolcsi for their helpful comments on this chapter.

some of these basic problems by introducing a variant of predicate logic that has a new language and a new class of models. The language is a propositional modal language; the models can be thought of as trees. The fact that the language is propositional and modal allows for a much closer fit with current Chomskian syntactic representations. The tree models allow for a direct, concrete semantics that does not involve the notion of an assignment.

In this dissertation I use the propositional modal language to specify logical form representations for a simple fragment of English. Because the fragment is small, it is important to argue that it is of real linguistic interest. To achieve this goal, the dissertation is composed of an overview chapter followed by four independent applications of the perspective to current issues in natural language semantics. Two of these applications are about pronominal anaphora: Chapter 2 is about dependent and independent pronouns and Chapter 3 is about static and dynamic binding. The other two applications are about linguistic applications of generalized quantifier theory: Chapter 4 is about reducibility to iterations and Chapter 5 is about conservativity and extension.

In the overview chapter, I present the perspective and illustrate the intuitions behind it. Section 2 introduces the informal intuition behind semantic trees. Section 3 defines the syntax of the modal language. Section 4 specifies logical form representations for a fragment of English. Section 5 introduces the tree models and uses them to define a semantics for the modal language. Section 6 discusses the core intuition about the connection between 'bound' pronouns and the copying operations that play the role of variables in the modal language. Section 7 gives a short description of each of the four application chapters.

In the case of the simple fragment in this dissertation, the connections between the four types of objects in (1) are simple: the modal formulas are close in form to standard syntactic representations; they have a standard semantics with respect to a class of intuitive models; and they are close enough to predicate logic formulas to make use of the latter's worked-out proof theory. Future research may tell how much of this simplicity can be maintained for larger fragments.

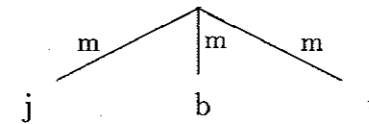
1.2 Semantic trees

The informal intuition behind semantic trees is that of a verification procedure, which takes a logical form representation and calculates its truth value in a predicate logic model by building a tree. For example, let \mathcal{D} be the following model: The individuals are John, Bill, and Tom, Susan, and Ruth; John loves Susan, Tom loves Ruth, Ruth loves Tom, and no one else loves anyone else.

The truth value of the sentence *every man loves some woman* in the model \mathcal{D} is calculated in the five steps (2)-(5).

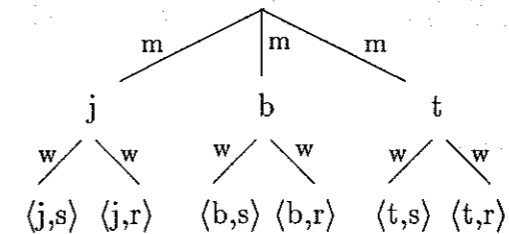
The first step is called an *extend* step. It extends the root with the individuals that are in the denotation of the predicate *man* in \mathcal{D} .

(2)



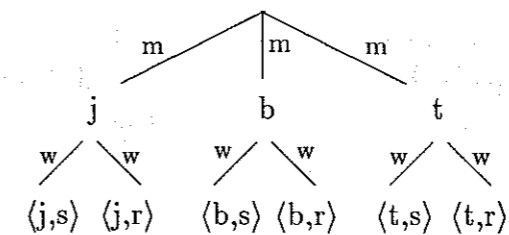
The second step is another *extend* step. It extends each individual with the individuals that are in the denotation of *woman* in \mathcal{D} .

(3)



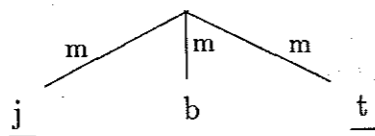
The third step is called a *mark* step. It marks those pairs that are in the denotation of *love* in \mathcal{D} .

(4)



The fourth step is called a *reduce* step. It applies the existential quantifier *some* to each individual. When applied to a node, the existential quantifier marks the node if at least one of its daughter nodes is marked, and deletes the daughter nodes.

(5)



The last step is another *reduce* step. It applies the universal quantifier *every* to the root. When applied to a node, the universal quantifier marks the node if every one of its daughter nodes is marked, and deletes the daughter nodes. At the end of the verification procedure, a marked root is interpreted as the truth value *true*, and an unmarked root as the truth value *false*. So the truth value of the sentence *Every man loves some woman* in the model \mathcal{D} is calculated to be *false*. The reader may wish to work out the similar steps in calculating that the truth value of the sentence *Some woman loves some man* in \mathcal{D} is *true*.

In general, common nouns function as *extenders*, verbs as *markers*, and determiners as *reducers*.

1.3 A modal language

In this section I introduce the syntax of the modal language \mathcal{L} that is used to specify logical form representations. This syntax will be specified in two steps. In the first, I give a concise formal definition of the language. In the second, I introduce abbreviations that make the language easier to read and use.

The definition in (6) defines what is a formula of the modal language:

$$(6) \phi ::= p \mid \delta_i \mid \neg\phi \mid \phi_1 \rightarrow \phi_2 \mid \Box\phi$$

The definition in (6) says the following. Basic formulas of the language are of two kinds: atomic formulas called *relation symbols*, and *delta predicates* of the form δ_i for some natural number i . Complex formulas are built from basic formulas by using negation, implication and the universal operator \Box . For example, if p and q are relation symbols, then p and δ_5 are formulas of \mathcal{L} , and so are $\neg q$ and $\Box(\delta_1 \rightarrow q)$.

Each relation symbol is associated with a natural number called its *type*. Intuitively, the type specifies the degree of the relation that will be the denotation of the relation symbol. For example, a relation symbol with type 2 will denote a binary relation, and a relation symbol with type 1 will denote a unary relation.

The first type of abbreviations define additional boolean operations and the existential operator \Diamond .

$$(7) \phi \vee \psi \stackrel{\text{def}}{=} \neg\phi \rightarrow \psi$$

$$\phi \wedge \psi \stackrel{\text{def}}{=} \neg(\neg\phi \vee \neg\psi)$$

$$\Diamond\phi \stackrel{\text{def}}{=} \neg\Box\neg\phi$$

For example, the last definition reflects the semantic intuition that *Some man came* is true if and only if *Every man did not come* is false.

The second type of abbreviations relate restricted and unrestricted quantification.

$$(8) \Box_\phi\psi \stackrel{\text{def}}{=} \Box(\phi \rightarrow \psi)$$

$$\Diamond_\phi\psi \stackrel{\text{def}}{=} \Diamond(\phi \wedge \psi)$$

For example, the first definition says that *Every man came* is true under the same conditions as *Every thing has the property that if it is a man then it came*. Similarly, the second definition, which can be derived from the first and the definitions in (7), says that *Some man came* is true under the same conditions as *Some thing has the property that it is a man and it came*.

With these abbreviations, we can think of \mathcal{L} as a *multi-modal* propositional language: A propositional modal language that has a family of operators of the form \Box_j for some set J of *restrictor symbols*, instead of the one operator \Box . For example, if q is a basic formula of a multi-modal propositional language \mathcal{L}' and j, k are restrictor symbols of \mathcal{L}' , then $\Box_j\Box_kq$ is a formula of \mathcal{L}' . In the special case where the members of J are themselves formulas of a propositional modal language, the abbreviations in (8) establish a correspondence between formulas of \mathcal{L}' and formulas of a single-modal language. For example, if δ_1 and q are formulas of \mathcal{L} , the \mathcal{L}' formula $\Box_{\delta_1}q$ corresponds to the \mathcal{L} formula $\Box(\delta_1 \rightarrow q)$.

The modal language \mathcal{L} is formally defined as a propositional language with a single \Box operator. In the rest of the overview chapter I keep the single modal perspective for formal definitions, but make use of the multi-modal perspective whenever it gives insight about \mathcal{L} and its models.

1.4 A fragment of English

In this section I specify logical form representations for a small fragment of English. This fragment will be extended in later sections and chapters. In this section, the fragment consists of transitive and intransitive verbs, and noun phrases built up from the determiners *every*, *some* and a lexical common noun. The syntactic representations of sentences with transitive and intransitive verbs

are assumed to be $[_{cp}DP[_{vp}V DP]]$ and $[_{cp}DP[_{vp}V]]$, respectively. This assumption too will be brought more in line with current Chomskian syntactic theory in Section 6. The definition in (9) defines a translation between these syntactic structures and the modal language defined in Section 3.

$$(9) \begin{aligned} [[[_{cp}DP VP]]] &= [[DP]][[VP]] \\ [[[_{vp}V DP]]] &= [[DP]][[V]] \\ [[[_{vp}V]]] &= [[V]] \\ [[[_{dp}D N]]] &= [[D]][[N]] \end{aligned}$$

Some lexical translations are:

$$(10) \begin{aligned} [[[_v love]]] &= love \\ [[[_v walk]]] &= walk \\ [[[_n man]]] &= man \\ [[[_n woman]]] &= woman \\ [[[_d every]]] &= \square \\ [[[_d some]]] &= \diamond \end{aligned}$$

In general, transitive verbs are translated as symbols for binary relations (relation symbols with type 2), intransitive verbs and common nouns are translated as symbols for unary relations (relation symbols with type 1), and determiners as symbols for modal operators.

The following is a sample derivation:

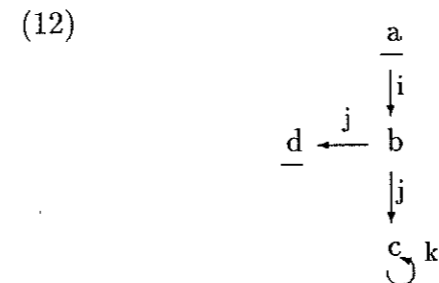
$$(11) \begin{aligned} &[[[_{cp}[_{dp}[_{d every}][_n man]][_{vp}[_v loves] [_{dp}[_{d some}][_n woman]]]]]] = \\ &[[[_{dp}[_{d every}][_n man]]]][[[_{vp}[_v loves] [_{dp}[_{d some}][_n woman]]]]] = \\ &[[[_{d every}]]][[[_n man]]]][[[_{dp}[_{d some}][_n woman]]]][[[_v loves]]] = \\ &\square man [[[_{d some}]]][[[_n woman]]] love = \\ &\square man \diamond woman love \end{aligned}$$

In this core fragment, the structure of the syntactic representations is very close to the structure of the logical form representations. In a sense, the translation amounts to thinking of verbs as relation symbols, determiners as modal operators and common nouns as restrictor symbols.

1.5 Tree models

In general, a model for a propositional modal language \mathcal{L} is a triple $\mathcal{M} = (S, R, V)$, where S is a non-empty set of objects called *points*, R is a binary relation on S called an *accessibility relation*, and V is a function called a *valuation* that assigns a subset of S to every basic formula of \mathcal{L} . A natural intuitive perspective is to think of each such model as a graph, where each member of R is an arrow that leads from some point to another, and each member of V is a 'color' that marks some of the points of the graph. Similarly, a model for a multi-modal propositional modal language is a triple $\mathcal{M} = (S, \{R_j\}_{j \in J}, V)$, where each operator of the form \square_j has its own accessibility relation R_j . In terms of the intuitive graph picture, one can think of each member of the accessibility relation R_j as an arrow with the label j .

For example, the picture in (12) depicts a model for a multi-modal language whose operator symbols are i, j , and k . The underline marks the points at which the relation symbol p is true.



In general, the truth definition for a propositional modal language \mathcal{L} defines the conditions under which a formula ϕ of \mathcal{L} is true at a point s in a model \mathcal{M} : A basic formula p is true at s if and only if s is in $V(p)$. Negation and implication are defined locally to each point, e.g., $\neg\phi$ is true at s if and only if ϕ is false at s . The modal operator \square express local quantification: $\square\phi$ is true at s if and only if ϕ is true at every point s' that is *accessible* from s by R , i.e., every point such that sRs' . For example, $\neg p$ is true at point c in the model in (12) because p is false at c . Consequently, $\square_k p$ is false at c because there is a point accessible from c by R_k , namely c , at which p is false. Consequently, $\square_j \square_k p$ is false at b .

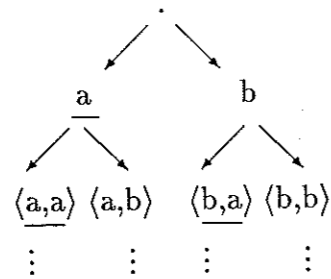
The formal truth definition for the modal language \mathcal{L} in Section 3 is in (13). The only non-standard clause is the clause for the delta predicates, which will not be discussed until the next section.

- (13) $\mathcal{M} \models_s p$ iff $s \in V(p)$
 $\mathcal{M} \models_s \delta_i$ iff $s \in V(\delta_i)$
 $\mathcal{M} \models_s \neg\phi$ iff $\mathcal{M} \not\models_s \phi$
 $\mathcal{M} \models_s \phi_1 \rightarrow \phi_2$ iff $\mathcal{M} \models_s \phi_1$ implies $\mathcal{M} \models_s \phi_2$
 $\mathcal{M} \models_s \Box\phi$ iff for all $s' \in S$, sRs' implies $\mathcal{M} \models_{s'} \phi$

In general, the points of models for \mathcal{L} are sequences. For the special purpose of specifying logical form representations for natural language, finite sequences seem enough. I will give an informal description of the models needed for the fragment, while a formal definition of the models needed for predicate logic can be found in the Appendix.

Crucially, models with finite sequences can be thought of as trees. And each tree model $\mathcal{M} = (S, R, V)$ can be thought of as corresponding to a familiar predicate logic model $\mathcal{D} = (D, I)$, where D is a non-empty set of individuals, and I is a function that assigns an n -ary relation over D to every relation symbol p with type n of the relevant predicate language. For example, the tree model depicted in (14) corresponds to a predicate logic model whose domain D consists of the individuals a and b , and p is a unary relation symbol such that $I(p)$ is $\{a\}$. The underline in (14) marks the points at which p is true as a relation symbol of the modal language \mathcal{L} .

(14)

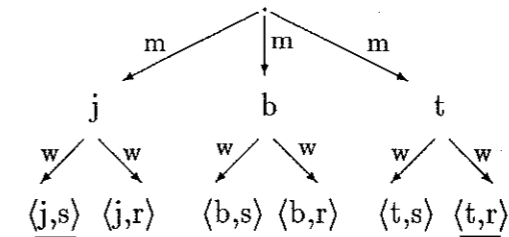


In general, the points of the tree model corresponding to \mathcal{D} are finite sequences of members of the domain D , each point is connected to all the points that are its extensions to the right by one member of D , and for every relation symbol p with type n , $V(p)$ is true of a sequence if and only if $I(p)$ is true of the last n members of the sequence.¹

¹A formal definition would need to address the question of sequences that are too short to evaluate a given basic formula. But this issue never arises in the interpretation of logical form representations for natural language, mainly because the Theta criterion ensures that each verb is evaluated at sequences that have been extended by the right number of noun phrases.

Every tree model \mathcal{M} has infinite depth, but in order to calculate the truth value of any specific formula ϕ of the modal language \mathcal{L} it is enough to consider a top part of \mathcal{M} with a finite depth, since every formula is finite and hence will have a finite depth of modal embedding. In addition, if ϕ is a multi-modal formula, only the relevant accessibility relations at each level of the tree need to be considered. For example, recall the model \mathcal{D} from Section 1. The individuals are John, Bill, and Tom, Susan, and Ruth; John loves Susan, Tom loves Ruth, Ruth loves Tom, and no one else loves anyone else. To calculate the truth value of the formula $\forall x(\text{man}(x) \rightarrow \exists y(\text{woman}(y) \wedge \text{love}(x,y)))$ in \mathcal{D} it is enough to evaluate the formula $\Box \text{man} \Diamond \text{woman loves}$ at the root of the tree in (15).

(15)



We can thus informally talk about a multi-modal formula and a predicate logic model determining a tree.

An important feature of the tree models is that they allow for an elegant solution to a basic technical problem in the application of Generalized Quantifier Theory to natural language. In Generalized Quantifier Theory, the meaning of a transitive verb is a *binary relation* while the meaning of a noun phrase is a set of *unary relations*, i.e., sets. The main obstacle to a simple treatment of transitive sentences is that the meaning of the object noun phrase has to combine with the binary relation that is the meaning of the verb. Tree models solve the problem by taking the elements of the models to be sequences: when a modal operator like \Box is associated with an object noun phrase, it quantifies over a *set* of points, but each point is itself a pair, i.e., a member of a binary relation. In other words, some of the combinatory burden involved in relating natural language to models is shifted to the models themselves.

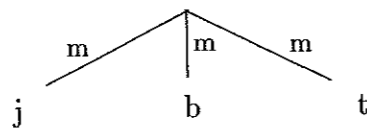
1.6 Pronouns and delta predicates

In the last section I introduced a semantics for the modal language in terms of a standard modal truth definition with respect to a new class of tree models. In this section I complete the description of the system with the introduction

of delta predicates and their truth definition. Intuitively, delta predicates are interpreted as 'copy operations' that compare the last element of a path to the element a fixed number of steps up the path. The informal verification procedure in Section 1 will be used to illustrate the truth definition for delta predicates via their role as translations for 'bound pronouns'. But in order to do that I will first extend the fragment in Section 3 to include *names*. All the trees in this chapter are determined by the model \mathcal{D} in Section 1.

As discussed in Section 1, a common noun can be thought of as an tree extender. For example, the tree in (16) results from the *extend* step of *man* as the common noun of a subject, when calculating the truth value of a sentences in \mathcal{D} .

(16)



By analogy, it is clear how to think of a name like a tree extender. For example, the tree in (17) results from the *extend* step of *John* as subject, when calculating the truth value of a sentences in \mathcal{D} .

(17)



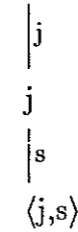
This intuition can be captured by extending the translation in Section 3 with the clause for names in (18), and lexical translations like the one in (19). This translation amounts to treating a name like *John* like a common noun *john* and an implicit universal determiner. Note, however, that since the relation symbol *john* denotes a set with exactly one member, any positive quantifier, for example \exists would do just as well. For example, *everyone who is John*, *someone who is John* and *the individual who is John* are true under the same conditions.

$$(18) \quad [[[_{dp} \text{NAME}]]] = [[\text{NAME}]]$$

$$(19) \quad [[[\text{John}]]] = \square_{\text{john}}$$

As was the case for other noun phrases, the same translation applies equally well to subjects and objects. For example, the translation of the sentence *John loves Susan* is $\square_{\text{john}} \square_{\text{susan}} \text{love}$. The tree determined by this formula and the model \mathcal{D} is (20). The formula is true at the root of the tree.

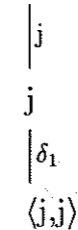
(20)



The tree in (20) captures the intuition that the truth of the sentence *John loves Susan* depends only on the value of the denotation of the relation symbol *love* at the pair $\langle j, s \rangle$.

In a similar way, the truth of a sentence like *John loves himself* depends only on the value of the denotation of the relation symbol *love* at the pair $\langle j, j \rangle$; but in terms of the informal verification procedure, this pair seems to be constructed in a different way than the pair in (20). Intuitively, the extend step associated with the reflexive pronoun *himself* in *John loves himself* copies the individual introduced by the extend step associated with the subject. The resulting tree is (21), where for every positive natural number i , the extend step of δ_i copies the individual i steps above it in the path.

(21)



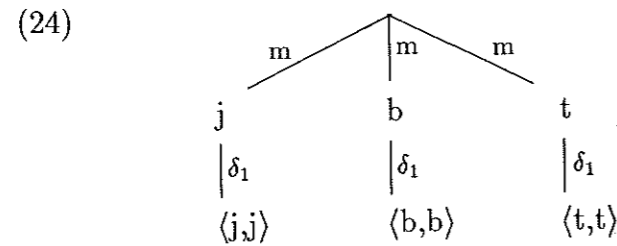
This intuition can be captured by extending the translation in Section 3 with the clause for pronouns in (22), and lexical translations like the one in (23). The choice of i in (23) is partly determined by grammatical constraints, like Binding theory in Chomskian syntax. Any remaining ambiguity, like all other forms of ambiguity, is not dealt with in this dissertation.

$$(22) \quad [[[_{dp} \text{PRONOUN}]]] = [[\text{PRONOUN}]]$$

$$(23) \quad [[[\text{himself}]]] = \square_{\delta_i}$$

The translation of the sentence *John loves himself* is $\Box_{john} \Box_{\delta_1} love$. The tree determined by this formula and the model \mathcal{D} is (21). The formula is false at the root of the tree.

The same translation works equally well when the antecedent of the pronoun is a quantified noun phrase like *every man*. For example, the translation of the sentence *Every man loves himself* is $\Box_{man} \Box_{\delta_1} loves$. The tree determined by this formula and the model \mathcal{D} is (24). The formula is false at the root of the tree.

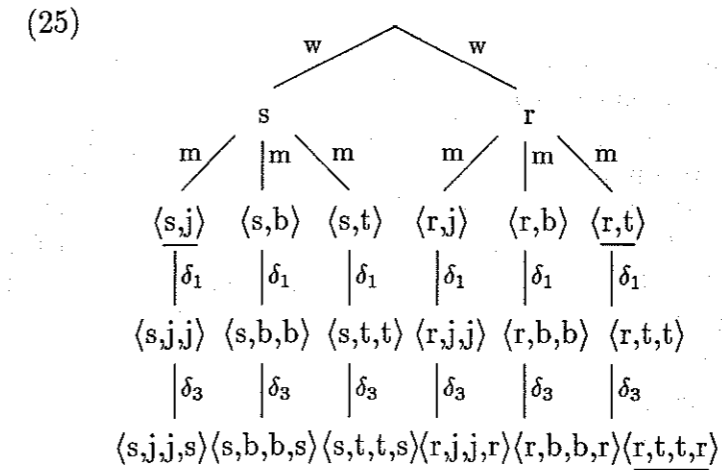


Formally, the accessibility relation R_{δ_i} connects a sequence s to the sequence s' that extends s to the right with its i -th element from the right. Thought about as a basic formula of \mathcal{L} , δ_i is true at a sequence s if and only if its last element is identical to its $i+1$ -th element from the right.

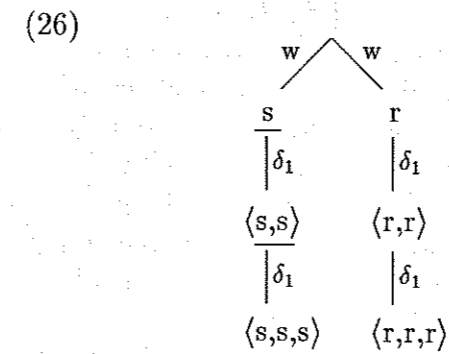
The idea of delta predicates has two important consequences.

Within logic itself, delta predicates allow the propositional language \mathcal{L} to express any predicate logic formula. A formal recursive translation definition can be found in the Appendix.

Within linguistics, delta predicates allow for a closer fit between logical form representations and the syntactic structures posited by current Chomskian syntax. As traces can be thought of as silent bound pronouns, it is easy to give logical form representations to structures with traces. For example, a structure like [*some woman_i [every man_j [t_j loves t_i]]]* can have the logical form representation $\Diamond woman \Box_{\delta_1} \Box_{\delta_3} love$. The tree determined by this formula and the model \mathcal{D} is (25). The formula is false at the root of the tree.



This example also illustrates the logical claim about the ability of \mathcal{L} to express any predicate logical formula. In particular structures with 'inverse scope' like the one above can be given a logical form representation. In addition, delta predicates make it easy to give logical form representations to structures with *intermediate* traces, which is something that is hard to do with predicate logic. For example, a structure like [*some woman_i [t_i [t_i walks]]]* can have the logical form representation $\Diamond woman \Box_{\delta_1} \Box_{\delta_1} walk$. The tree determined by this formula and a model \mathcal{D}' that is like \mathcal{D} but in addition specifies that Susan walks but Ruth does not, is the one in (26). The formula is true at the root of the tree.



1.7 A road map

The rest of the dissertation are four chapters, which are four independent applications of different aspects of the perspective introduced in this chapter to issues in natural language semantics.

Chapter 2, *Dependent and independent pronouns* is about the distinction between dependent (bound anaphora, sloppy) pronouns and independent (coreference, strict) pronouns, exemplified by the ambiguity in (27).

(27) John loves his mother, and Bill does too.

The second sentence in (27) can mean that Bill loves Bill's mother, or that he loves John's mother. The first reading corresponds to a dependent reading of the pronoun in the first sentence; the second reading corresponds to an independent reading of the pronoun in the first sentence.

Using the *fragment*, dependent pronouns are translated as delta predicates, and independent pronouns are translated as name-like relation symbols. This approach can be seen as a conceptually simpler and more general way of expressing the basic insights of Fiengo and May (1994). In particular, translating anaphors, bound pronouns and traces as copy operators suggests a new perspective on the connection between anaphora and movement.

Chapter 3, *Static and dynamic binding*, is about the relation between static binding, exemplified in (28), and dynamic binding, exemplified in (29).

(28) There is a woman that every man likes.

(29) There is a man. He walks.

The covert pronoun in *There is a woman that every man likes (her)* is bound within the scope of the existential quantifier; the pronoun in *He walks* is bound outside the scope of the existential quantifier.

Using the *modal language*, the chapter presents a simple system that combines the system of semantic trees and the system of predicate logic with anaphora of Dekker (1994). In the combined system, both static and dynamic binding are expressed with delta predicates. One natural hypothesis about the semantic difference between them is the following: A statically bound delta predicate targets entities in an initial set (considered at the beginning of quantification); A dynamically bound delta predicate targets entities in a final subset (found at the end of quantification).

Chapter 4, *Generalized quantifier reducibility*, a reprint of Ben-Shalom (1994a), deals with the formal question of when a generalized quantifier can be expressed by an iteration of generalized quantifiers of lower types. This question is of linguistic interest because it is a measure of the interdependence between the denotations of noun phrases within a sentence:

(30) Every man loves some woman.

(31) Different men love different women.

The binary quantifier that corresponds to the subject-object pair in (30) is called *reducible* because it is equal to the iteration of the unary quantifiers EVERY MAN and SOME WOMAN. The binary quantifier that corresponds to the subject-object pair in (31) is called *irreducible* because it is not equal to the iteration of any two unary quantifiers.

Using the *tree models*, the chapter proves all the unreducibility results in Keenan (1992) and other results, using a general graphic proof technique based on a characterization of reducibility to iterations as invariance under simple transformations on tree models.

Chapter 5, *Conservativity and extension*, out of Ben-Shalom (1994b), deals with the generalized quantifier conditions of Conservativity (32) and Extension (33), which are generally assumed to be true of all natural language quantifiers.

(32) $Q_{MAB} \Leftrightarrow Q_{MA} A \cap B$

(33) if $A, B \subseteq M \subseteq M'$ then
 $Q_{MAB} \Leftrightarrow Q_{M'AB}$

For example, (32) says that the interpretation of *every man laughed* does not depend on laughers who are not men; and (33) says that the interpretation of *every man laughed* does not depend on things that are neither men nor laughers; together, they say that the interpretation of *every man laughed* depends only on men.

Using the *connection between generalized quantifiers and modal operators*, this idea of *domain restriction* (van Benthem (1984)) is shown to correspond to the basic modal invariance under generated submodels.

Appendix

A Language

Let \mathcal{L} be a propositional modal language, whose set Q of *atomic formulas* is the union of two disjoint sets: A denumerable set Φ of *relation symbols*, and the set Δ of *delta predicates* of the form δ_i , one for each positive $i \in \omega$.

Each $q \in Q$ is associated with a non-negative integer called its *type*. The type of every $\delta_i \in \Delta$ is fixed to be 1.

A typical member of Φ is denoted by p , and when no confusion arises, it is assumed to have type j .

The formulas of \mathcal{L} are then defined as usual, by:

$$\phi ::= \perp \mid p \mid \delta_i \mid \phi_1 \rightarrow \phi_2 \mid \Box \phi$$

B Models

A model for \mathcal{L} is a triple $\mathcal{M} = (S, R, V)$, which meets the following conditions:
 A is a non-empty set.
 S is the set of sequences of length ω over A . I.e.,

$$s \in S \text{ iff } s = \dots s_k \dots s_1 s_0, \quad s_i \in A \text{ for all } i \in \omega$$

R is the binary relation on S defined by:

$$sRs' \text{ iff } s' = sa \text{ for some } a \in A$$

For every $p \in \Phi$ of type j , P is an j -ary relation over A .
 V is the function from Q to subsets of S defined by:

$$s \in V(q) \text{ iff } \begin{cases} s_{j-1} \dots s_1 s_0 \in P & \text{if } q = p \\ s_i = s_0 & \text{if } q = \delta_i \end{cases}$$

Satisfaction in \mathcal{M} is then defined as usual, by:

$$\begin{aligned} \mathcal{M} \models_s q & \quad \text{iff } s \in V(q) \\ \mathcal{M} \not\models_s \perp & \\ \mathcal{M} \models_s \phi_1 \rightarrow \phi_2 & \quad \text{iff } \mathcal{M} \models_s \phi_1 \text{ implies } \mathcal{M} \models_s \phi_2 \\ \mathcal{M} \models_s \Box \phi & \quad \text{iff for all } s' \in S, sRs' \text{ implies } \mathcal{M} \models_{s'} \phi \end{aligned}$$

C Translation

Definition 1 A subformula ϕ of ψ is at modal depth i in ψ if it is in the scope of exactly i \Box operators.

Definition 2 A subformula δ_i of ψ is free in ψ if it is at modal depth at most i . Otherwise, it is bound.

Definition 3 The formula $[+1]\phi$ is obtained from ϕ by replacing each δ_i free in ϕ with δ_{i+1} .

Definition 4 The formula $[-1]\phi$ is obtained from ϕ by replacing each δ_i free in ϕ with δ_{i-1} .

Definition 5 The formula ϕ_i^t is obtained from ϕ by replacing each $[+1]^i t$ at modal depth i in ϕ with $[+1]^i t'$.

Let Φ be a set of relation symbols, each with a non-negative integer called its type.

Let A be a non-empty set and for each $p \in \Phi$ of type j , let P be a j -ary relation over A .

Let \mathcal{L}_f be the first-order language determined by Φ .

Let \mathcal{L}_m be the modal model determined by Φ , as above.

Let \mathcal{M}_f be the \mathcal{L}_f -model determined by A and $\{P \mid p \in \Phi\}$.

Let \mathcal{M}_m be the \mathcal{L}_m -model determined by A and $\{P \mid p \in \Phi\}$, as above.

We define the following translations between \mathcal{L}_f and \mathcal{L}_m :

$$(\perp)^m = \perp$$

$$(p(x_{i_1} x_{i_2} \dots x_{i_j}))^m = \Box([+1]\delta_{i_1} \rightarrow \Box([+1]^2 \delta_{i_2} \rightarrow \dots \Box([+1]^j \delta_{i_j} \rightarrow p) \dots))$$

j times

$$(x_k = x_i)^m = \Box([+1]\delta_k \rightarrow [+1]\delta_i)$$

$$(\phi_1 \rightarrow \phi_2)^m = (\phi_1)^m \rightarrow (\phi_2)^m$$

$$(\forall x_k \phi)^m = \Box([+1]((\phi)^m)_{\delta_0}^{[+1]\delta_k})$$

$$(\perp)^f = \perp$$

$$(p)^f = p(x_{j-1} \dots x_1 x_0)$$

$$(\delta_i)^f = x_i = x_0$$

$$(\phi_1 \rightarrow \phi_2)^f = (\phi_1)^f \rightarrow (\phi_2)^f$$

$$(\Box \phi)^f = \forall x_{\text{new}} ([-1](((\phi)^f)_{[+1]x_{\text{new}}}^{\Box}))$$

Proposition 1 For every $s \in A^\omega$, $\phi_m \in \mathcal{L}_m$, $\phi_f \in \mathcal{L}_f$,

$$\mathcal{M}_m \models_s \phi_m \text{ iff } \mathcal{M}_f \models \phi_m^f[s]$$

$$\mathcal{M}_m \models_s \phi_m^m \text{ iff } \mathcal{M}_f \models \phi_f[s]$$

References

- Ben-Shalom, D. 1994a. A Tree Characterization of Generalized Quantifier Reducibility. In Makoto Kanazawa and Christopher J. Piñón, eds., *Dynamics Polarity, and Quantification*. Stanford: CSLI.
- Ben-Shalom, D. 1994b. Natural Language, Generalized Quantifiers, and Modal Logic. In *Proceedings of the 9th Amsterdam Colloquium*. University of Amsterdam.
- Benthem, J.v. 1984. Questions about Quantifiers, *Journal of Symbolic Logic* 49, 443 – 466.
- Dekker, P. 1994. Predicate logic with Anaphora. In Lynn Santeland and Mandy Harvy, eds., *Proceedings of the 4th Semantics and Linguistic Theory Conference*. Cornell University, Ithaca, NY.
- Fiengo, R. and R. May. 1994. *Indices and Identity*. MIT Press, Cambridge, Mass.
- Keenan, E. 1992. Beyond the Frege Boundary. *Linguistics and Philosophy* 15, 199 – 221.
- Stabler, E. to appear. Computing Quantifier Scope. In Anna Szabolcsi, ed. *ways of scope taking*. Kluwer.

Chapter 2

Dependent and Independent Pronouns*

2.1 Introduction

This chapter is about the distinction between dependent and independent pronouns. This distinction, also known as bound anaphora vs. coreference or sloppy vs. strict, has received a fair amount of attention within generative grammar. Although often treated as a syntactic issue, the following semantic intuition is relatively uncontroversial: the reference of an independent pronoun is established independently, while the reference of a dependent pronoun is established indirectly, via a linguistic antecedent. In this chapter I show that this intuition can be expressed in a version of predicate logic that uses copy operations instead of variables. Using this language to express logical forms, independent pronouns are translated as name-like operators, while dependent pronouns are translated as copy operators that indicate the semantic address of their antecedents.

Logical forms expressed with this language have a simple direct semantics in terms of semantic trees. At the same time, and largely because the copy operations allow for a direct interpretation of traces, these logical forms are very similar in structure to traditional LF representations. From one perspective, this chapter is an illustration of the usefulness of the new logical forms for linguistic theory. From another, its treatment of the distinction between dependent and independent pronouns can be seen as a particularly simple way of executing the basic insights of Fiengo and May (1994): the name-like operators correspond to their α - occurrences and the copy operators to their β - occurrences. Yet more generally, it suggests a new perspective on the old idea about the connection between anaphora and movement: anaphors, bound pronouns and traces are all copy operators.

Section 2 introduces the intuitive distinction between independent and dependent pronouns. Section 3 reviews Reinhart's (1983) treatment of this distinction in terms of coreference vs. binding. Section 4 summarizes Fiengo and May (1994) treatment of the distinction in terms of α - vs. β -occurrences.

*I would like to thank Tanya Reinhart, Ed Stabler, and Anna Szabolcsi for their helpful comments on this chapter.

Section 5 presents a treatment of the distinction in terms of name-like vs. copy operators. Section 6 discusses the empirical problems with Fiengo and May's extension of their core idea to independent β - and dependent α -occurrences. Section 7 shows that a key result of Fiengo and May can be seen as a minimality condition on dependent pronouns which is not unlike familiar minimality conditions on traces.

2.2 Independent vs. dependent pronouns

The basic intuition behind the distinction between independent and dependent pronouns ascribes an ambiguity to simple sentences like (1).

- (1) John loves his mother.

According to one reading, where the pronoun is *independent*, John loves John's mother. According to another, where the pronoun is *dependent*, John loves his own mother. What makes the distinction elusive is that the two readings have the same truth conditions. Where they differ is in the way the pronoun *his* picks its reference: either independently, like a name; or in a dependent way, via a linguistic antecedent. This distinction is easier to justify in the context of verb phrase ellipsis, where the two readings of the pronoun give rise to different truth conditions.

- (2) John loves his mother, and Bill does too.

Under the assumption that the interpretation of the second verb phrase is determined by that of the first, the claim about the ambiguity of (1) accounts for the two relevant readings of (2). If the pronoun is independent in the first sentence, it will determine the same reference in the second, leading to an interpretation in which Bill loves John's mother. If the pronoun is dependent in the first sentence, it will determine the same dependency in the second, leading to an interpretation in which Bill loves Bill's mother.

2.3 Coreference vs. binding

One influential theory of the distinction between independent and dependent occurrences of pronouns is that of Reinhart (1983). According to this theory, independent occurrences of pronouns involve *coreference*, while dependent occurrences involve *binding*. Of these two notions, only binding is linguistically determined: a noun phrase is bound if it is coindexed with a *c*-commanding

noun phrase. Bound pronouns are interpreted as bound variables, while the interpretation of pronouns that are not bound is determined pragmatically: two noun phrases that are not coindexed can still end up as corefering, i.e., referring to the same entity, subject to pragmatic considerations of the speaker's intentions.

For example, the independent reading of (1) has the syntactic representation in (3), and the dependent reading has the representation in (4).

- (3) John_i loves his_i mother.

- (4) John_i loves his_i mother.

These syntactic representations are interpreted semantically as in (5) and (6), respectively. Crucially, a bound pronoun is assumed to correspond to a variable, that is bound within a predicate formed by lambda abstraction.

- (5) j loves j 's mother

- (6) $\lambda x[x$ loves x 's mother] (j)

The main conceptual problem with Reinhart's (1983) theory is the lack of distinction between semantic and pragmatic coreference. This distinction is illustrated by the sentence in (7), due to Higginbotham (1985).

- (7) He put on John's coat.

The relevant reading is the one in which *he* corefers with *John*. This particular reading can only be felicitously used in a special pragmatic context, say in a party where the speaker intends to *imply* that *he* refers to *John* without asserting it. A natural way of describing this reading is to say that it has the semantic representation in (8)

- (8) k put on j 's coat

and that in the intended context, both *k* and *j* refer to *John*. This description is not available within Reinhart's theory.

The main technical problem with Reinhart's theory concerns the role of lambda abstraction. Basically, this use of abstraction entails an inherent mismatch between syntactic and semantic representations. Specifically, if the only direct syntactic counterparts of variables are pronouns and traces, then the only principled way of relating syntactic structures like (4) and semantic structures like (6) is by movement. Concretely, the subject in (6) would have to be moved out of its surface position so that its trace can be interpreted as a variable. More generally, every antecedent of a pronoun may have to be moved at LF just in order to create a variable between itself and the pronoun it binds.

2.4 α - vs. β -occurrences

A recent theory of the distinction between independent and dependent occurrences of pronouns is that of Fiengo and May (1994). According to this theory, every occurrence of a pronoun has an index composed of a numerical *indexical value* and an *indexical type* that is either α or β . In the core cases, independent occurrences of pronouns are α -occurrences while dependent occurrences are β -occurrences.

For example, the independent reading of (1) has the syntactic representation in (9), and the dependent reading has the representation in (10).

(9) John₁ ^{α} loves his₁ ^{α} mother.

(10) John₁ ^{α} loves his₁ ^{β} mother.

Fiengo and May specify a semantics for a small fragment of English that includes sentences (9) and (10). It is stated in terms of the semantic value of expressions with respect to sequences of individuals. In symbols, $\text{Val}(\chi, \Phi, \sigma)$ reads as χ is the value of expression Φ with respect to the sequence σ . The interpretations of the values of α - and β -occurrences are defined by (11) and (12).

(11) $\text{Val}(x, [_{np} + \text{pronoun}]_i^\alpha, \sigma)$ iff $x = \sigma(i)$

(12) $\text{Val}(x, [_{np} + \text{pronoun}]_i^\beta, \sigma)$ iff $\text{Val}(x, \text{NP}_i, \sigma)$

In other words, while the value of an α -occurrence of a pronoun is obtained directly from a sequence, the value of a β -occurrence of a pronoun is resolved by substituting a noun phrase with the same indexical value. The reason substitution yields the right truth conditions is that Fiengo and May assume that every quantified noun antecedent is moved at LF.

The basic insight of Fiengo and May (1994) treatment of independent and dependent occurrences of pronouns is that while independent occurrences do not depend on linguistic structure, dependent occurrences do. This difference can be demonstrated in the context of verb phrase ellipsis, where an overt and a covert verb phrase have to be similar enough for the structure to be well formed. The core case is (13).

(13) John loves his mother, and Bill thinks that Tom does too.

Assuming that the first overt verb phrase in (13) claims that John loves John's mother, the covert verb phrase can claim that Tom loves John's mother or that Tom loves Tom's mother. But it cannot claim that Tom loves Bill's

mother, despite the fact that *Bill* c-commands the covert verb phrase in (13). Intuitively, the difference between the first two readings and the third is the following. The first reading involves two independent pronouns that are similar enough by virtue of referring to the same person, namely John. The second reading involves two dependent pronouns that are similar enough by virtue of depending on an antecedent that has the same syntactic position relative to them, namely the subject of their own clause. The third reading involves two dependent pronouns that are not similar enough because they depend on antecedents that have different syntactic positions relative to them, namely the subject of their own clause and the subject of the clause above them, respectively.

Fiengo and May (1994) execute this idea in terms of α - and β -occurrences, by positing that the three readings above have the three representations in (14)-(16).

(14) John₁ ^{α} loves his₁ ^{α} mother and Bill₂ ^{α} thinks that Tom₃ ^{α} loves his₁ ^{α} mother

(15) John₁ ^{α} loves his₁ ^{β} mother and Bill₂ ^{α} thinks that Tom₃ ^{α} loves his₃ ^{β} mother

(16) * John₁ ^{α} loves his₁ ^{β} mother and Bill₂ ^{α} thinks that Tom₃ ^{α} loves his₂ ^{β} mother

In (14) the overt and covert verb phrases are similar enough because the two α -occurrences of the pronouns have the same *value*, namely 1. In (15) the overt and covert verb phrases are similar enough because the two β -occurrences of the pronouns have the same *dependency*, namely the one in (17). In (16) the overt and covert verb phrases are not similar enough because the two β -occurrences of the pronouns have different dependencies, namely the ones in (18a) and (18b), respectively. Structural dependencies between pronouns and their antecedents are specified by a mechanism that involves Chomsky (1955) style linear structural descriptions of phrase markers. The complex details of this mechanism can be ignored for the present purposes.

(17) $\langle \text{NP}, \text{V}, \text{NP} \rangle$

(18a) $\langle \text{NP}, \text{V}, \text{NP} \rangle$

(18b) $\langle \text{NP}, \text{V}, \text{NP}, \text{V}, \text{NP} \rangle$

The main technical problem with this theory is that the pronouns in the overt and covert verb phrases cannot be required to be identical. Rather, their similarity is defined in indirect and complex ways. Some empirical problems with this theory will be discussed in Section 6.

2.5 Names vs. delta predicates

Fiengo and May (1994) argue that the distinction between independent and dependent occurrences of pronouns has a semantic basis: the value of an independent occurrence of a pronoun is established directly, while that of a dependent occurrence is established indirectly, from the value associated with another noun phrase in the sentence. In this section we suggest that this semantic intuition can be executed directly, in logical forms where pronouns are represented as operators that specify explicitly in what way they establish their value. Concretely, a singular independent pronoun can be represented as a name, or any other operator that refers to a fixed individual irrespective of syntactic structure; a singular dependent pronoun can be represented as a copy operator that specifies the semantic address of its antecedent. These copy operators, called *delta predicates* are introduced and defined in the semantic trees framework of Chapter 1. In this chapter they are used on an intuitive basis.

The idea behind semantic trees is that to calculate the truth value of a sentence in a given model, one evaluates a formula determined by the logical form of the sentence in a tree model determined conjointly by the formula and the model. For example, consider the sentence *Some woman loves every man*. This sentence has a reading according to which for every man there is a woman that loves him. According to current theories of logical form, this reading is represented as something like (19), where irrelevant details have been suppressed.

$$(19) [\text{every man}_i [\text{some woman}_j [t_j \text{ loves } t_i]]]$$

The logical form in (19) determines the modal formula in (20), where asymmetric syntactic *c-command* translates into the scope of modal operators. \Box is the universal modal operator, and \Diamond is the existential modal operator.¹

$$(20) [\Box \text{ man} [\Diamond \text{ woman} [\Box_{\delta_1} \Box_{\delta_3} \text{ love}]]]$$

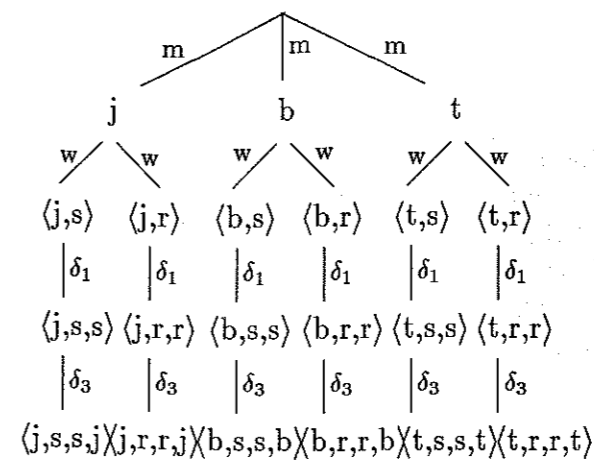
Now imagine the the model in (21).

$$(21) \text{ The men are John, Bill and Tom, the women are Susan and Ruth; Susan loves John, Ruth loves Tom, and no one else loves anyone else.}$$

The semantic tree determined by the formula in (20) and the model in (21) is the tree in (22). The underlined sequences are the ones where the second individual from the right loves the rightmost one.

¹Since a singular pronoun or a name quantifies over exactly one individual, it makes no difference whether its modal operator is taken as \Box or \Diamond .

(22)



As can be seen by inspecting the sequences at the bottom of the tree in (22), the formula in (20) is false in the model in (21), because not every man m has a woman w such that w loves m . The general idea is that noun phrases are evaluated according to their *c-command* order, thereby creating the right sequences for the verb to be evaluated at. More concretely, the operator associated with each noun phrase extends every sequence at the bottom of the tree with the individuals that are relevant for the evaluation of the noun phrase. For example, the operator \Box_{man} corresponding to *every man* extends every sequence with all the men in the model. Similarly, the operator \Box_{john} corresponding to *John* extends every sequence with the individual John. In other words, singular pronouns are always associated with operators of the form \Box_{ind} . These operators are like names in that they extend every sequence with a single individual. But this individual can be determined in more than one way. In particular, it can be determined by the structure of the tree itself: an operator of the form \Box_{δ_i} extends each sequence with its i -th member from the right. In other words, it is a copy operator.

The independent reading of (1) has the logical form in (23), and the dependent reading has the logical form in (24).

$$(23) [\Box_{\text{john}} [\Box_{[\Box_{\text{john}}]} \text{ mother} [\text{ love}]]]$$

$$(24) [\Box_{\text{john}} [\Box_{[\Box_{\delta_2}]} \text{ mother} [\text{ love}]]]$$

To derive just the right readings for the verb phrase ellipsis in (13) it is enough to assume that the covert verb phrase has an identical logical form to the overt one. Assuming that the overt verb phrase in (13) claims that John loves John's mother, the covert verb phrase has the two logical forms in (25) and (26).

(25) $[\Box_{[\Box_{john}] mother [love]}]$

(26) $[\Box_{[\Box_{b_2}] mother [love]}]$

When combined with the representation of the embedded subject *Tom*, the embedded sentence in (13) claims that Tom loves John's mother, according to (25), or that Tom loves Tom's mother, according to (26). The unavailable reading (16), according to which Tom loves Bill's mother, is unavailable here, without further stipulation. The logical forms in this section have a standard and simple semantics as propositional modal formulas. The details are spelled out in Chapter 1.

2.6 Dependent α - and independent β - occurrences

In the last section we argued that the core distinction between independent and dependent occurrences of pronouns should be viewed as a distinction between name-like operators and copy operators. From this general operator perspective, these two options are just two of the ways in which the operator corresponding to a singular pronoun can pick its individual. This plurality contrasts with Fiengo and May (1994) distinction between α - and β -occurrences of pronouns, which is a strict dichotomy. One could then expect a principled source of empirical problems for Fiengo and May's theory: namely, whenever the core distinction between independent α - occurrences and dependent β -occurrences is stretched to describe occurrences of pronouns that fall properly outside the core dichotomy. In this section we discuss two such cases, which can be described as independent β -occurrences, and dependent α -occurrences, respectively.

Recall from Section 4 that in the core cases, α -occurrences are considered identical if they have the same indexical *value*, while β -occurrences are considered identical if they exhibit the same *dependency*. But Fiengo and May chose to compromise this distinction by allowing β -occurrences to be considered identical if they have the same *value*. This is done in order to account for the verb phrase ellipsis in sentences like (27).

(27) Max thinks he is strong, Oscar does too, but his father doesn't.

The relevant reading is the one according to which Max thinks that Max is strong, Oscar thinks that Oscar is strong and Oscar's father thinks that Oscar is strong. Fiengo and May represent this reading by representation in (28).

(28) Max_1 thinks he_1^β is strong, $Oscar_2$ thinks he_2^β is strong, but his_2^α father doesn't think he_2^β is strong

To maintain that the embedded verb phrases in (28) can be considered identical, Fiengo and May claim that the first and second occurrences of the pronoun *he* are identical by virtue of having the same dependency, the second and third are identical by having the same value, and so all three are identical by transitivity.

But these assumptions entail a direct incorrect prediction with respect to the sentence in (29).

(29) His father thinks that he is strong, Oscar does too, but Max doesn't.

Since the embedded verb phrases are required to be considered identical but no order restrictions are involved, the representation in (30) should be well-formed: the second and third occurrences of the pronoun *he* are identical by virtue of having the same dependency, and the first and third are identical by having the same value.² But (29) clearly cannot mean that Max's father thinks that Max is strong, Oscar thinks that Oscar is strong and Max doesn't think that Max is strong.

(30) His_2^α father think that he_2^β is strong, $Oscar_1$ thinks that he_1^β is strong, but Max_2 doesn't think that he_2^β is strong

The second empirical problem with Fiengo and May (1994) theory is the mirror image of the first, in a sense; whereas the conceptual source of trouble in the first is that the indexical value of β -occurrences mattered, the conceptual source of trouble in the second is that the indexical value of α -occurrences do not matter, in the sense that they are used for quantification. This is done in order to account for the verb phrase ellipsis in sentences like (31).

(31) I know which book Max read, and which book Sally thinks that Oscar did.

This sentence is given the representation in (32).

²That (30) is not ill-formed because of the backward anaphora is shown by sentences like *His father thinks that he is strong, and Max does too*.

- (32) I know [which book₁ [Max read e₁^α] and [which book₁ [Sally thinks that [Oscar read e₁^α]]]

To maintain that the most embedded verb phrases in (32) can be considered identical, Fiengo and May claim that the first and second occurrences of the trace *e* are identical by virtue of having the same value. Intuitively, this amounts to treating the trace as a free variable that can get caught by any operator with the same indexical value that has scope over it.

This assumption entails a direct incorrect prediction with respect to the sentence in (33).

- (33) I know which girl Max insulted, and which boy told which girl that Oscar did.

Since the only relevant requirement is that the most embedded verb phrases can be considered identical, the representation in (34) should be well-formed.³ So (33) should mean something like *I know which girl Max insulted, and which boy told which girl that Oscar insulted her*. But it clearly doesn't.⁴

- (34) I know [which girl₁ [Max insulted e₁^α] and [which boy [told which girl₁ that [Oscar insulted e₁^α]]]

2.7 Minimality

A key result for Fiengo and May (1994) is their ability to account for the following observation: if both pronouns in the first sentence of (35) refer to Max, the second sentence in (35) has the three readings in (36a)-(36c) but not the reading in (36d).

- (35) Max said he saw his mother, and Oscar did, too.

³That (34) is not ill-formed because the ellided pronoun corresponds to a trace in the first sentence and to an overt noun phrase in the second is shown by sentences like *Dulles suspected Philby, who Angleton did too*.

⁴That there is c-command between *which girl* and the pronoun in the second sentence in (34) is shown by the Condition C violation in *I told him that Max is stupid*.

- (36a) Oscar said Max saw Max's mother
 (b) Oscar said Oscar saw Oscar's mother
 (c) Oscar said Oscar saw Max's mother
 (d) * and Oscar said Max saw Oscar's mother

Adding more pronouns in the first sentence, the following generalization is observed: once any of the pronouns in the ellided verb phrase refers to Max, subsequent pronouns in the ellided verb phrase have to refer to Max too.

For Fiengo and May, the generalization is the following: once any of the pronouns in the ellided verb phrase is an independent occurrence, subsequent pronouns in the ellided verb phrase are independent occurrences too. For example, the logical forms of the readings in (36) are schematically described as follows:

- (37a) $\alpha \alpha$
 (b) $\beta \beta$
 (c) $\beta \alpha$
 (d) * $\alpha \beta$

Their account of this generalization hinges on particular details of their mechanism of factorization. For example, the missing reading in (36d) corresponds to the logical form in (38).

- (38) Max₁ said he₁^α saw his₁^β mother, and Oscar₂ said he₁^α saw his₂^β mother

A crucial component of the reason why (38) is not a possible reading for (35) is that the dependency in (39) is not realized in the first sentence of (35).

- (39) $\langle\langle\text{Max, his}\rangle, 1, \langle\text{NP, V, NP}\rangle\rangle$

But this is only so because the definition of realizing a dependency stipulates that all pronouns with the same indexical value in the scope of a dependency have to be members of the factorization.

In the present chapter, dependent pronouns are thought of as a special case of copy operators, along with anaphors and traces; this suggests a different perspective on the generalization above. Assume that all the pronouns in the first sentence in (35) can be either dependent or independent. In other words, all the options in (40) are available for the pronouns in the *first* sentence of (35).

(40a) i i

(b) d d

(c) d i

(d) i d

To exclude the reading in (36d), it is enough to assume the following condition on dependent readings: a dependent pronoun has to copy its value from the closest possible noun phrase. The options in (40) thus correspond to the following schematic logical forms for the first sentence:⁵

(41a) *max max max*

(b) *max δ_1 δ_1*

(c) *max δ_1 max*

(d) *max max δ_1*

Copied as the interpretation of the ellided verb phrase, these logical forms determine the following logical forms for the second sentence:

(42a) *oscar max max*

(b) *oscar δ_1 δ_1*

(c) *oscar δ_1 max*

(d) *oscar max δ_1*

The logical forms in (42a)-(42c) give the readings in (36a)-(36c), respectively; but the logical form in (42d) does not give the missing reading in (36d); rather, it gives another way of deriving the reading in (36a).

An important advantage of this solution is that the minimality condition it assumes is of a familiar kind, once dependent pronouns and traces are both thought of as copy operators. It is just a version of the familiar condition that minimizes the chain links between traces and their antecedents.

⁵Ignoring the operators for the noun phrase *his mother* and, possibly, for the embedded clause itself.

2.8 Conclusion

This chapter is about the distinction between dependent and independent pronouns. It suggests that the distinction can be captured by logical forms where independent pronouns are translated as name-like operators that pick their reference directly, whereas dependent pronouns are translated as copy operators that establish their reference via an antecedent noun phrase. This approach is argued to be conceptually superior to other approaches to the distinction, in particular the coreference vs. binding approach of Reinhart (1983) and the α -occurrences vs. β -occurrences approach of Fiengo and May (1994). In particular, copy operators suggest a new perspective on the connection between anaphora and movement.

References

- Chomsky, N. 1955/1975. *The Logical Structure of Linguistic Theory*. Plenum Press, New York.
- Fiengo, R. and R. May. 1994. *Indices and Identity*. MIT Press, Cambridge, Mass.
- Higginbotham, J. 1985. On Semantics. *Linguistics Inquiry*. 16, 547 – 594.
- Reinhart, T. 1983. *Anaphora and Semantic Interpretation*. Croom Helm, London.

Chapter 3

Static and Dynamic Binding*

3.1 Introduction

This chapter is about the similarities and differences between static ('bound anaphora') and dynamic binding. A key difference between statically and dynamically bound variables is that the former but not the latter are bound by operators that have syntactic scope over them. Among the two main types of approaches to dynamic binding, the so-called E-type pronoun approach (Evans (1985)) treats dynamic binding completely separately from ordinary predicate logic 'static' binding of variables. The so-called bound anaphora approach, on the other hand, (Groenendijk and Stokhof (1991)) treats dynamic binding as involving the binding of variables, thus making it similar to ordinary predicate logic 'static' binding. Recently, Dekker (1994) suggested a successor to Groenendijk and Stokhof's dynamic predicate logic which keeps variables for static binding but eliminates them from dynamic binding. His main argument is that the 'syntactically free but somehow semantically bound' variables used to achieve dynamic binding create technical and conceptual complications that can be eliminated by replacing variables with anaphoric terms. In his system, static and dynamic binding are incomparable again. Independently, Ben-Shalom (1994) showed that predicate logic itself can be thought of as a propositional modal system that involves copy predicates rather than variables. Both Dekker's anaphoric terms and Ben-Shalom's copy predicates can be thought of as copy operations based on paths. This chapter suggests a synthesis of the two systems, where static and dynamic binding are both expressed by copy predicates. In the combined system, the difference between static and dynamic binding correlates with a semantic distinction: static binding targets entities in an initial set (considered at the beginning of quantification); dynamic binding targets entities in a final subset (found at the end of quantification).

Technically, it is thus quite possible to have a simple unified representation for static and dynamic binding within a variant of dynamic predicate logic. The conceptual issues deserve further consideration in future work.

*I would like to thank Nissim Francez and Ed Stabler for their helpful comments on this chapter.

Section 2 summarizes the semantic trees system of Chapter 1. Section 3 summarizes the predicate logic with anaphora system of Dekker (1994). Section 4 presents a system that combines the two. Section 5 uses the combined system for a semantic comparison of static and dynamic binding.

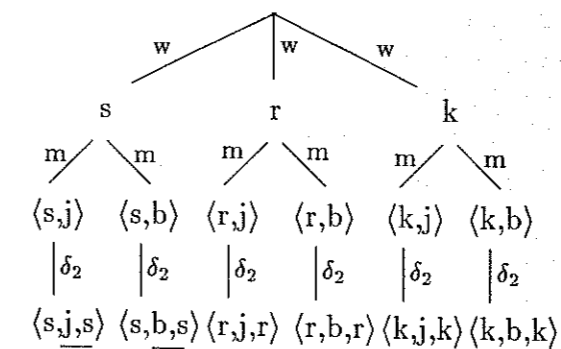
3.2 Semantic trees

The semantic trees system (Chapter 1) is a version of predicate logic whose models can be thought of as trees. The truth value of a predicate logic sentence in a predicate logic model \mathcal{D} can be determined by evaluating the appropriate propositional modal formula at the root of a tree. The tree has depth n for some finite n , and it corresponds to an n -ary relation over \mathcal{D} . Formulas are evaluated as tuples: For example, the binary predicate ADMIRE is true at a tuple e iff the second member of e from the right admires the first member of e from the right according to \mathcal{D} . A statically bound pronoun is translated as a copy predicate δ_i for an appropriate i . The copy predicate δ_i is true of a tuple e iff the first member of e from the right is identical to the $i + 1$ -th member of e from the right.

The following example illustrates how ST works. The simplest way to read the formula below is as a translation of *There is a woman that every man admires (her)*. The tree is the part of \mathcal{M} that is relevant for evaluating this formula in the model \mathcal{M} determined by $D = \{j, b, s, r, k\}$ with $MAN = \{j, b\}$, $WOMAN = \{s, r, k\}$ and $ADMIRE = \{\langle j, s \rangle, \langle b, s \rangle, \langle b, k \rangle\}$. The formula is true at the root of the tree.

(1) There is a woman that every man admires.

$$\diamond(woman \wedge \Box(man \rightarrow \diamond(\delta_2 \wedge admire)))$$



Formally, the semantic trees system is defined as follows: The ST language is a propositional modal language. Its set of atomic formulas

is the disjoint union of two sets: a set Φ of *relation symbols*, each with a natural number called its arity; and a set $\Delta = \{\delta_i \mid i \in \omega\}$ of *delta predicates*.

The formulas of ST are defined as follows:

Definition 1 (syntax of ST)

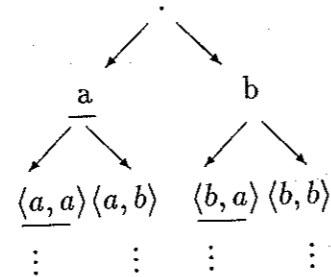
$$\phi ::= p \mid \delta_i \mid \neg\phi \mid \phi \wedge \psi \mid \diamond\phi$$

with $p \in \Phi$, $\delta_i \in \Delta$. As usual, $\Box\phi$ and $\phi \rightarrow \psi$ abbreviate $\neg\diamond\neg\phi$ and $\neg(\phi \wedge \neg\psi)$, respectively.

Complex formulas are built up from atomic formulas by negation, conjunction and existential quantification. The only difference from the syntax of ordinary propositional modal logic is the additional set of atomic formulas Δ .

An ST model $\mathcal{M} = \langle S, R, V \rangle$ is determined by a non-empty domain D of individuals, and a set of finitary relations over D with a set P of n -ary tuples of individuals for each relation symbol p of arity n . S is the set of tuples of finite length over D . sRs' iff $s' = s \cdot d$ for some $d \in D$. $s \cdot \langle d_0, \dots, d_{n-1} \rangle \in V(p)$ iff n is the arity of p and $\langle d_0, \dots, d_{n-1} \rangle \in P$. $s \cdot \langle d_0, \dots, d_{i+1} \rangle \in V(\delta_i)$ iff $d_0 = d_{i+1}$.

This definition is illustrated in the figure below, which depicts a model \mathcal{M} for an ST language with one relation symbol q , of arity 1, where $D = \{a, b\}$, and $Q = \{a\}$. The underlined tuples are the ones in $V(q)$.



If defined, the truth value of a formula ϕ of ST at a tuple s in a model \mathcal{M} is determined in the ordinary propositional modal logic way. The only reason for the truth value of ϕ to be undefined at a state s is if it contains an atomic formula q evaluated at a tuple s'' which is too short for it: either q is a relation symbol of arity n and $|s''| < n$, or q is the delta predicate δ_i and $|s''| \leq i + 1$

Definition 2 (semantics of ST)

$$\begin{aligned} \mathcal{M} \models_s p & \text{ iff } s \in V(p) \\ \mathcal{M} \models_s \delta_i & \text{ iff } s \in V(\delta_i) \\ \mathcal{M} \models_s \neg\phi & \text{ iff } \mathcal{M} \not\models_s \phi \\ \mathcal{M} \models_s \phi \wedge \psi & \text{ iff } \mathcal{M} \models_s \phi \text{ and } \mathcal{M} \models_s \psi \\ \mathcal{M} \models_s \diamond\phi & \text{ iff } \mathcal{M} \models_{s'} \phi \text{ for some } s', sRs' \end{aligned}$$

3.3 Predicate logic with anaphora

The predicate logic with anaphora system (Dekker (1994)) is a version of dynamic predicate logic whose information states can be thought of as states of knowledge about n roles for some finite n . An information state with degree n about a predicate logic model \mathcal{D} corresponds to an n -ary relation over \mathcal{D} . For example, for every predicate logic \mathcal{D} with domain D , the minimal state of knowledge about n roles is the full set of tuples D^n : the n roles can be played by any tuple of n individuals. The processing of a formula in an information state σ with degree n can change σ in two ways: it can rule out certain tuples of individuals in σ as playing these n roles; and it can extend tuples in σ to tuples in a new state σ' with more than n roles. A dynamically bound pronoun is translated as an anaphoric term p_i for an appropriate i . For each tuple e , the anaphoric term p_i 'copies' the i -th member of e from the right.

The following example illustrates how PLA works:

(2) There is a man. He walks.

$$\exists x M(x) \wedge W(p_0)$$

$$\begin{aligned} \sigma[[\exists x M(x)]]_{\mathcal{M},g} &= \{e \cdot d \mid d \in D \wedge e \in \sigma[[M(x)]]_{\mathcal{M},g[x/d]}\} \\ &= \{e \cdot d \mid e \in \sigma \wedge d \text{ is a man}\} (= \sigma') \\ \sigma'[[W(p_0)]]_{\mathcal{M},g} &= \{e' \in \sigma' \mid \text{the last element of } e' \text{ walks}\} \\ &= \{e \cdot d \mid e \in \sigma \wedge d \text{ is a man} \wedge d \text{ walks}\} \end{aligned}$$

Formally, the predicate logic with anaphora system is defined as follows: The PLA language is constructed from sets of relation constants R^n of arity n , a set C of individual constants, and countable sets V and $A = \{p_i \mid i \in \omega\}$ of variables and pronouns, respectively. The sets C , V and A together constitute the set of terms T .

The formulas of PLA are defined as follows:

Definition 3 (syntax of PLA)

$$\phi ::= R(t_1, \dots, t_n) \mid t_1 = t_2 \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \exists x\phi$$

with $t_i \in T$, $R \in R^n$, $x \in V$. As usual, $\forall x\phi$ and $\phi \rightarrow \psi$ abbreviate $\neg\exists x\neg\phi$ and $\neg(\phi \wedge \neg\psi)$, respectively.

A basic formula is either a relation symbol and the right number of terms, or an equality. Complex formulas are built up from basic formulas by negation, conjunction and existential quantification. The only difference from the syntax of ordinary predicate logic is the additional set of anaphor terms A .

A PLA model $\mathcal{M} = \langle D, F \rangle$ is a non-empty domain D of individuals, and an interpretation function F which assigns individuals in D to individual constants and sets of n -tuples of individuals to relations constants of arity n . An information state σ about \mathcal{M} is a relation of degree k over D for some $k \geq 0$; if σ is non-empty its degree $|\sigma|$ is the length of the tuples in σ ; if σ is empty its degree $|\sigma|$ is inherent. Information states are ordered by a partial order, based on a partial order on tuples: $e \leq e'$ iff there is a tuple e'' such that $e' = e \cdot e''$. $\sigma \leq \sigma'$ iff the degree $|\sigma|$ of σ is no larger than the degree $|\sigma'|$ of σ' , and for every tuple e' in σ' there is a tuple e in σ such that $e \leq e'$. This definition is illustrated in the figure below:

$$\left\{ \begin{array}{c} \langle d_1, \dots, d_n \rangle \\ \vdots \\ \langle d'_1, \dots, d'_n \rangle \\ \vdots \\ \langle d''_1, \dots, d''_n \rangle \\ \vdots \\ \langle d'''_1, \dots, d'''_n \rangle \end{array} \right\} \leq \left\{ \begin{array}{c} \langle d'_1, \dots, d'_n, d''_{n+1}, \dots, d''_{n+m} \rangle \\ \vdots \\ \langle d''_1, \dots, d''_n, d'''_{n+1}, \dots, d'''_{n+m} \rangle \end{array} \right\}$$

Individual constants and variables are evaluated as in ordinary predicate logic with respect to a model and an assignment function, respectively. Pronouns are evaluated with respect to a case $e = \langle e_1, \dots, e_{|e|} \rangle$ of an information state σ .

Definition 4

$$\begin{aligned} [c]_{\mathcal{M}, \sigma, e, g} &= F(c) \\ [x]_{\mathcal{M}, \sigma, e, g} &= g(x) \\ [p_i]_{\mathcal{M}, \sigma, e, g} &= e_{|e|-i} \quad (\text{if } |\sigma| > i) \end{aligned}$$

If defined, the dynamic interpretation $\sigma[[\phi]]_{\mathcal{M}, g}$ of a PLA formula ϕ in an information state σ is a state σ' such that $\sigma \leq \sigma'$. The formula that follows ϕ is interpreted in the updated state σ' . The only reason for the dynamic interpretation of ϕ to be undefined at a state σ is if it contains a pronoun p_i that cannot be evaluated at a state σ'' : either $|\sigma''| \leq i$ or σ'' is empty.

Definition 5 (semantics of PLA)

$$\begin{aligned} \sigma[[R(t_1, \dots, t_n)]]_{\mathcal{M}, g} &= \{e \in \sigma \mid \langle [t_1]_{\mathcal{M}, \sigma, e, g}, \dots, [t_n]_{\mathcal{M}, \sigma, e, g} \rangle \in F(R)\} \\ & \quad (\text{if } |\sigma| > i \text{ for every } p_i \in \{t_1, \dots, t_n\}) \\ \sigma[[t_1 = t_2]]_{\mathcal{M}, g} &= \{e \in \sigma \mid [t_1]_{\mathcal{M}, \sigma, e, g} = [t_2]_{\mathcal{M}, \sigma, e, g}\} \\ & \quad (\text{if } |\sigma| > i \text{ for every } p_i \in \{t_1, t_2\}) \\ \sigma[[\neg\phi]]_{\mathcal{M}, g} &= \{e \in \sigma \mid \neg\exists e' \text{ such that } e \leq e' \wedge e' \in \sigma[[\phi]]_{\mathcal{M}, g}\} \\ \sigma[[\phi_1 \wedge \phi_2]]_{\mathcal{M}, g} &= \sigma[[\phi_1]]_{\mathcal{M}, g} \cap \sigma[[\phi_2]]_{\mathcal{M}, g} \\ \sigma[[\exists x\phi]]_{\mathcal{M}, g} &= \{e \cdot d \mid d \in D \wedge e \in \sigma[[\phi]]_{\mathcal{M}, g[x/d]}\} \end{aligned}$$

where $|\sigma[[R(t_1, \dots, t_n)]]_{\mathcal{M}, g}|, |\sigma[[t_1 = t_2]]_{\mathcal{M}, g}|, |\sigma[[\neg\phi]]_{\mathcal{M}, g}| \stackrel{\text{def}}{=} |\sigma|$, and $|\sigma[[\exists x\phi]]_{\mathcal{M}, g}| \stackrel{\text{def}}{=} |\sigma| + 1$.

3.4 A combined system

This section combines the PLA and ST systems. In the combined system, relations of finite degree are used for both information states and the evaluation of formulas. Both statically and dynamically bound pronouns are translated as copy predicates. The syntax of the combined system is as simple as that of ST, and its semantics is no more complex than that of PLA.

The following example illustrates how CS works:¹

(3) There is a man. He walks.

$\diamond man; \diamond(\delta_0; walk)$

$$\begin{aligned} \sigma[[\diamond man]]_{\mathcal{M}} &= \{e' \mid eRe' \text{ for some } e \in \sigma\}[[man]]_{\mathcal{M}} \\ &= \{e \cdot d \mid e \in \sigma \wedge d \text{ is a man}\} (= \sigma') \\ \sigma'[[\diamond(\delta_0; walk)]]_{\mathcal{M}} &= \{e'' \mid e'Re'' \text{ for some } e' \in \sigma'\}[[\delta_0]]_{\mathcal{M}}[[walk]]_{\mathcal{M}} \\ &= \{e' \cdot d' \mid e' \in \sigma' \wedge \\ & \quad d' \text{ is equal to the last element of } e' \wedge d' \text{ walks}\} \\ &= \{e \cdot d \cdot d' \mid e \in \sigma \wedge d \text{ is a man} \wedge d' \text{ walks}\} \end{aligned}$$

Formally, the predicate logic with anaphora system is defined as follows: The syntax of the CS language is practically identical to that of ST:

¹If ST 'individual terms' are preferred not to have dynamic effects, they can be treated as involving \square rather than \diamond .

Definition 6 (syntax of CS)

$$\phi ::= p \mid \delta_i \mid \sim\phi \mid \phi; \psi \mid \diamond\phi$$

with $p \in \Phi$, $\delta_i \in \Delta$. As usual, $\Box\phi$ and $\phi \rightarrow \psi$ abbreviate $\sim\diamond\sim\phi$ and $\sim(\phi; \sim\psi)$, respectively.

A CS model is just an ST model $\mathcal{M} = \langle S, R, V \rangle$, as defined in Section 2. An information state σ about \mathcal{M} is just a PLA information state about the base set D of \mathcal{M} , as defined in Section 3.² If defined, the dynamic interpretation of a CS formula ϕ at an information state σ about a model \mathcal{M} is defined very much as in PLA. The only reason for the dynamic interpretation of a formula ϕ to be undefined at a state σ is if ϕ contains an atomic formula q evaluated at a state σ'' whose tuples are too short for q as an ST formula, as defined in Section 2. In the following definition, $R^* \stackrel{\text{def}}{=} \bigcup_{i \in \omega} R^i$.

Definition 7 (semantics of CS)

$$\begin{aligned} \sigma[[p]]_{\mathcal{M}} &= \{e \in \sigma \mid e \in V(p)\} \\ \sigma[[\delta_i]]_{\mathcal{M}} &= \{e \in \sigma \mid e \in V(\delta_i)\} \\ \sigma[[\sim\phi]]_{\mathcal{M}} &= \{e \in \sigma \mid \neg\exists e' \text{ such that } eR^*e' \wedge e' \in \sigma[[\phi]]_{\mathcal{M}}\} \\ \sigma[[\phi_1; \phi_2]]_{\mathcal{M}} &= \sigma[[\phi_1]]_{\mathcal{M}}[[\phi_2]]_{\mathcal{M}} \\ \sigma[[\diamond\phi]]_{\mathcal{M}} &= \{e' \mid eR^*e' \text{ for some } e \in \sigma\}[[\phi]]_{\mathcal{M}} \end{aligned}$$

where $|\sigma[[p]]_{\mathcal{M}}|, |\sigma[[\delta_i]]_{\mathcal{M}}|, |\sigma[[\sim\phi]]_{\mathcal{M}}| \stackrel{\text{def}}{=} |\sigma|$, and $|\sigma[[\diamond\phi]]_{\mathcal{M}}| \stackrel{\text{def}}{=} |\sigma| + 1$.

3.5 Static and dynamic binding

But CS is more than an elegant way of combining the systems of semantic trees and predicate logic with anaphora. Because it uses copy predicates to express both static and dynamic binding it offers a simple way of comparing them. A natural perspective about CS takes each tuple in an information state as a path, and interpretation as a process that eliminates and/or extends paths. In terms of this perspective, the two types of binding are similar in that both involve a copy operator that targets entities a fixed number of steps up every path. In addition, the interpretation of a quantified formula of the form $\diamond(\phi)$ seem to involve the following steps: an initial set S of entities is considered at the point $\diamond(\uparrow)$; some of these entities are eliminated during the interpretation of ϕ , until a final subset S' of S is left at the point $\diamond(\phi)^\downarrow$. In

²Information states were defined with respect to $\mathcal{M} = (D, F)$, but in fact only use D .

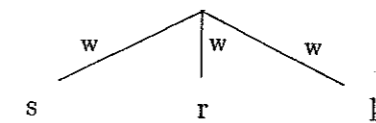
terms of this perspective, a statically bound copy operator seems to target entities in an initial set S while a dynamically bound copy operator seems to target entities in a final set S' . This distinction is illustrated in the following example, which depicts the current information state σ at several points during the processing of a formula with one statically bound delta predicate and one dynamically bound delta predicate. For simplicity, the initial information state is taken to be $\langle \rangle$, the minimal information state about 0 roles. The formula in (4) is evaluated in this information state with respect to the model determined by $D = \{j, b, s, r, k\}$, $\text{MAN} = \{j, b\}$, $\text{WOMAN} = \{s, r, k\}$, $\text{ADMIRE} = \{\langle j, s \rangle, \langle b, s \rangle, \langle b, k \rangle\}$, and $\text{BEAUTIFUL} = \{s, r\}$.

(4) There is a woman that every man admires. She is beautiful.

$$\diamond(\text{woman}; \Box(\text{man} \rightarrow \diamond(\delta_2; \text{admire}))); \diamond(\delta_0; \text{beautiful})$$

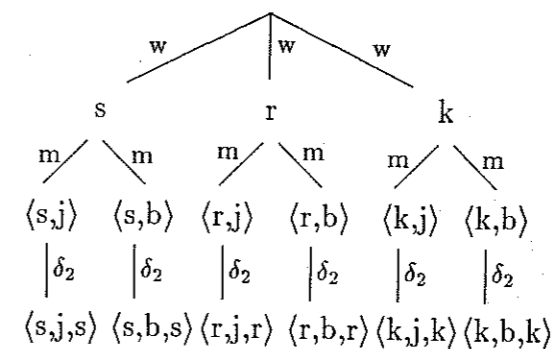
- $\diamond(\text{woman}^\downarrow)$

The first quantification considers the set S of the women in D .



- $\diamond(\text{woman}; \Box(\text{man} \rightarrow \diamond(\delta_2^\downarrow))$

The statically bound δ_2 targets the entities in S .



- $\diamond(woman; \Box(man \rightarrow \diamond(\delta_2; admire)))^\dagger$
The woman s is the only entity in S that meets the condition that every man admires her. S' is the set $\{s\}$.

$$\begin{array}{c} | \\ w \\ | \\ s \end{array}$$

- $\diamond(woman; \Box(man \rightarrow \diamond(\delta_2; admire))); \diamond(\delta_0)^\dagger$
The dynamically bound δ_0 targets the entities in S' .

$$\begin{array}{c} | \\ w \\ | \\ s \\ | \\ \delta_0 \\ | \\ \langle s, s \rangle \end{array}$$

3.6 Conclusion

This chapter presents a system that combines the predicate logic with anaphora system of Dekker (1994) with the semantic trees system of Chapter 1. In the combined system both static and dynamic binding are expressed by copy operations based on paths. The difference between the two types of binding correlates with a semantic distinction: static binding targets entities in an initial set (considered at the beginning of quantification); dynamic binding targets entities in a final subset (found at the end of quantification).

References

- Ben-Shalom, D. 1994. *A Path-based Variable-free System for Predicate Logic*. Technical Report CS-R9444. CWI, Amsterdam.
- Dekker, P. 1994. Predicate Logic with Anaphora. In Lynn Santeland and Mandy Harvy, eds., *Proceedings of the 4th Semantics and Linguistic Theory Conference*. Cornell University, Ithaca, NY.
- Evans, G. 1985. *Collected Papers*. Foris, Dordrecht.
- Groenendijk, J. and M. Stokhof. 1991. Dynamic Predicate Logic. *Linguistics and Philosophy* 14, 39 – 100.

Chapter 4

Generalized Quantifier Reducibility*

4.1 Introduction

Already in Montague Grammar, sentences like *Some cat sneezed* were analyzed as involving Generalized Quantifiers: the Verb Phrase *sneezed* denotes the property SNEEZE, i.e., the set of entities who sneezed; The subject Noun Phrase *some cat* denotes the generalized quantifier SOME CAT, i.e., the set of properties that include at least one cat. The truth value of the sentence *Some cat sneezed* is true iff the property SNEEZE is in the set SOME CAT, i.e., if there is at least one cat who sneezed.

This type of analysis had to be extended in order to interpret transitive sentences like *Every dog bit some cat*. The crucial difference is that at least one of the Noun Phrases has to apply to the binary relation BITE rather than to a property. A natural way of doing this, suggested in Keenan (1987), is to extend the domain of the generalized quantifier SOME CAT so it includes binary relations in addition to properties. The value of SOME CAT at the binary relation BITE is then defined as the property of having bitten some cat, i.e., $\{x : \{y : x \text{ BITE } y\} \in \text{SOME CAT}\}$. With this extension of the domain of SOME CAT, the interpretation of *Every dog bit some cat* is straightforward: the Verb Phrase *bite some cat* denotes the property SOME CAT(BITE), i.e., the set of entities that bit some cat. The subject Noun Phrase *every dog* denotes the (unextended) generalized quantifier EVERY DOG, and the sentence *Every dog bit some cat* is true iff the property BITE SOME CAT is in the set EVERY DOG, i.e., if for every dog there is a cat which that dog bit.

This analysis of transitive sentences is attractive for current linguistic theories for two reasons. First, in a sense it is a minimal extension of the generalized quantifiers analysis of intransitive sentences. The quantification is analysed in a form like $Q_1(Q_2(R))$, where the generalized quantifier Q_1 is the interpretation of the subject, the extended generalized quantifier Q_2 is the interpretation of the object and the binary relation R is the interpretation of the transitive verb. In other words, the only additional assumption is the simple extension of

*I would like to thank Ed Keenan, Johan van Benthem, Dag Westerståhl, and an anonymous reviewer for their helpful comments concerning this paper.

the domain of generalized quantifiers. Second, the analysis is strictly compositional, i.e., the semantic interpretation of a transitive sentence can be carried out in a way that respects the syntactic structure of the sentence. In particular, the interpretation of the sentence is derived from the interpretation of its two immediate syntactic constituents, namely the Verb Phrase and the subject Noun Phrase.

But not all transitive sentences are amenable to such an analysis. For example, it is not obvious how to derive the interpretation of the sentence *Different people like different things* from the interpretation of the Verb Phrase *like different things* and the interpretation of the subject *different people*. Formally, the binary quantifier defined by the interpretation of the subject-object pair DIFFERENT PEOPLE_DIFFERENT THINGS is not *reducible*: there is no generalized quantifier Q_1 and extended generalized quantifier Q_2 such that for all binary relations R DIFFERENT PEOPLE_DIFFERENT THINGS(R) = $Q_1(Q_2(R))$.

Keenan (1987,1992) and van Benthem (1989) present natural language binary quantifiers that are not reducible, and offer formal tests to determine whether a given binary quantifier is reducible or not. Van Benthem's test is easy to apply, but it is only applicable to a subset of natural language quantifiers, namely the permutation invariant ones. Keenan's tests are Reducibility Equivalence (RE) and Reducibility Characterization (RC). RE is not general. RC is general but it is hard to apply: using RC to prove that a given binary quantifier Q_0 is unreducible amounts to proving that for every extended generalized quantifier Q_2 , there is no generalized quantifier Q_1 such that $Q_0 = Q_1 \circ Q_2$. But one is not told how to prove this.

The Graphic Invariance criterion developed in this paper can be used to construct simple, visual and uniform unreducibility proofs. More importantly, the invariance perspective makes it clear what makes a quantifier unreducible. Keenan and van Benthem consider a variety of English expressions which determine unreducible quantifiers, but it is unclear from their works whether these quantifiers have anything in common. What Graphic Invariance says is that a reducible quantifier is invariant to certain graphic transformations. In other words, an unreducible quantifier is unreducible because it distinguishes between relations that 'look the same' to reducible quantifiers.

Finally and perhaps most importantly, the current literature on unreducible quantifiers deals exclusively with the special case of binary quantifiers. Consequently, it has little to say about natural language quantifiers that are determined by sentences involving a subject, an object, and an indirect object. For example, the quantifiers determined by sentences such as *Different teachers assigned different questions to different students* or *Two guests introduced themselves to each other*.

The more general and conceptually simpler proof technique developed in this paper makes it possible to show that the two sentences above determine unreducible ternary quantifiers. In addition, it enables one to discriminate between different patterns of unreducibility in sentences with three Noun Phrases: sentences like *Two guests introduced themselves to each other*, in which the dependency between the Noun Phrases spans the subject and both of the objects; sentences like *Mary introduced each guest to his host* in which the dependency spans the object and the indirect object only, but not the subject; and sentences like *Each guest introduced his host to Mary*, in which it spans the subject and the object, but not the indirect object.

The paper is organized as follows:

In Section 2, I give the basic definitions: an n -ary relation, an (extended) Generalized Quantifier (GQ) of type $\langle n \rangle$, and type $\langle k \rangle$ reducibility.

In Section 3, I state and prove a generalization of Keenan's RC, and characterize reducible GQs in terms of Graphic Invariance, a graphic formulation of RC, based on a tree interpretation of n -ary relations. Using Graphic Invariance, I develop an easy, visual and general proof technique for proving that a given quantifier is not type $\langle k \rangle$ reducible.

In Section 4, I use the Graphic Invariance proof technique to show that certain natural language GQs are not type $\langle k \rangle$ reducible. I cover all of Keenan's core examples of unreducible binary GQs, as well as some closely related cases whose unreducibility is hard to prove by his tests. Finally, I discuss unreducible ternary GQs.

4.2 Definitions

4.2.1 Relations over \mathcal{E}

Let us write \mathcal{E} for the set (assumed non-empty) of objects under discussion. An n -ary relation over \mathcal{E} is a set of n -tuples of elements of \mathcal{E} . I write the set of all n -ary relations over \mathcal{E} as \mathcal{R}^n . The truth values *true* and *false* are treated as the universal and empty 0-place relations, respectively.

4.2.2 GQs of type $\langle n \rangle$

We define,

- (1) F is an (extended) *generalized quantifier* (GQ) of type $\langle n \rangle$ iff F extends a GQ from \mathcal{R}^n to $\{0, 1\}$ in the following way:
For all $R \in \mathcal{R}^{k+n}$, $k > 0$,

$$F(R) = \{ \langle a_1, \dots, a_k \rangle : \\ F(\{ \langle b_1, \dots, b_n \rangle : \langle a_1, \dots, a_k, b_1, \dots, b_n \rangle \in \mathcal{R} \}) = 1 \}$$

For example, the value of the extended GQ SOME CAT at the binary relation BITE is $\{x : \text{SOME CAT}(\{y : x \text{ BITE } y\}) = 1\}$, i.e., the property of having bitten some cat.

A extended GQ of type $\langle n \rangle$ is completely determined by the GQ it extends. So it can be characterized in terms of its values on n -ary relations. For example, the subject-object pair in both (2a) and (2b) are interpreted as GQs of type $\langle 2 \rangle$. In (2a) it is the GQ that checks whether a binary relation contains the tuples $\langle \text{john}, \text{john} \rangle$ and $\langle \text{mary}, \text{mary} \rangle$. In (2b) it is the GQ that checks whether every dog stands in the relevant relation to some cat.

- (2) a. John and Mary (both) love themselves.
b. Every dog bit some cat.

4.2.3 Type $\langle k \rangle$ reducibility

A GQ of type $\langle n \rangle$ is called type $\langle k \rangle$ reducible iff it is the composition of two GQs of types $n - k$ and k , $n > k$.

We define,

- (3) A GQ F of type $\langle n \rangle$ is type $\langle k \rangle$ reducible iff there is a GQ f of type $\langle n - k \rangle$ and a GQ g of type $\langle k \rangle$ such that $F = f \circ g$.

For example, the binary quantifier EVERY DOG_SOME CAT determined by the sentence *Every dog bit some cat* is type $\langle 1 \rangle$ reducible, because it is equal to EVERY DOG \circ SOME CAT. Here $F = \text{EVERY DOG_SOME CAT}$, $f = \text{EVERY DOG}$ and $g = \text{SOME CAT}$.

Note that a type $\langle n \rangle$ quantifier that is type $\langle k \rangle$ reducible does not have to be type $\langle m \rangle$ reducible for any other m , $0 < m < n$. For example, we prove in Section 4.3 that the type $\langle 3 \rangle$ quantifier determined by the sentence *Mary introduced each guest to his host* is type $\langle 2 \rangle$ but not type $\langle 1 \rangle$ reducible. Similarly, the type $\langle 3 \rangle$ quantifier determined by the sentence *Each guest introduced his guest to Mary* is type $\langle 1 \rangle$ but not type $\langle 2 \rangle$ reducible. Finally, one technical remark. In practice, it is more convenient to use the definition of type $\langle k \rangle$ reducibility given in (4), in which g is known to be positive, i.e., true only of sets that are not empty. For example MORE THAN FOUR CATS is positive, while AT MOST FOUR CATS is not. Generalizing a proof by Keenan (1992), it can be shown that the definitions in (3) and (4) are equivalent. Intuitively, this is true because of equivalences like the one

between *Every dog bit at most four cats* and *No dog bit more than four cats*. Formally, $f \circ g = f - \circ - g$, and one of $g, -g$ is positive, where for all R , $(-g)(R) =_{def} \neg(g(R))$ and $(f-)(R) =_{def} f(\neg(R))$.

- (4) A GQ F of type $\langle n \rangle$ is type $\langle k \rangle$ reducible iff there is a GQ f of type $\langle n - k \rangle$ and a positive GQ g of type $\langle k \rangle$ such that $F = f \circ g$.

4.3 Graphic Invariance

The type $\langle k \rangle$ reducible type $\langle n \rangle$ GQs can be characterized by a simple generalization of Keenan's Reducibility Characterization (RC) theorem. The generalized RC can in principle be used to prove that a certain GQ is not type $\langle k \rangle$ reducible. But since it does not suggest any specific proof technique, it is in practice quite hard to apply.

An alternative formulation of RC is in terms of Graphic Invariance, based on a natural representation of n -ary relations as trees: A quantifier F is $f \circ g$ for some quantifier g iff F is invariant to certain graphic transformations induced by g . Graphic Invariance can be used to construct simple, visual and uniform unreducibility proofs.

4.3.1 Type $\langle k \rangle$ Reducibility Characterization

The theorem in (5) characterizes the type $\langle k \rangle$ reducible type $\langle n \rangle$ GQs. It is proved by a simple generalization of Keenan's original proof for the special case $k = 1, n = 2$.

(5) Type $\langle k \rangle$ Reducibility Characterization

Let F be a GQ of type $\langle n \rangle$. F is type $\langle k \rangle$ reducible iff \exists a positive GQ g of type $\langle k \rangle$, $k < n$, s.t. for all n -ary relations R, R' $F(R) = F(R')$ if $g(R) = g(R')$.

Proof:

$$\implies : F(R) = f \circ g(R) = f(g(R)) = f(g(R')) = f \circ g(R') = F(R')$$

\impliedby : Define a GQ f of type $\langle n - k \rangle$ by: $f(S) = 1$ iff $\exists R'$ s.t. $F(R') = 1$ and $g(R') = S$. Then $f(g(R)) = 1$ iff $\exists R'$ s.t. $F(R') = 1$ and $g(R') = g(R)$ iff $F(R) = 1$. So $F = f \circ g$.

4.3.2 A tree interpretation of n -ary relations

(6) Let $R \in \mathcal{R}^n$. We define $T(R)$, or *the tree determined by R* , as follows:

$$T(R) =_{\text{def}} (N_R, \preceq),$$

where the set N_R of *nodes* is the set of prefixes of members of R :

$$N_R = \{ \langle a_1, \dots, a_k \rangle : 0 \leq k \leq n \text{ and}$$

$$\exists a_{k+1}, \dots, a_n \text{ s.t. } \langle a_1, \dots, a_k, a_{k+1}, \dots, a_n \rangle \in R \}$$

and the domination relation \preceq is the prefix relation:

$$\alpha \preceq \beta \text{ iff } \exists k, m \ 0 \leq k \leq m \leq n \text{ s.t.}$$

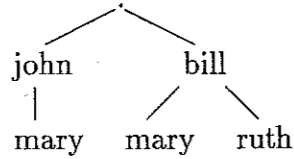
$$\alpha = \langle a_1, \dots, a_k \rangle \text{ and } \beta = \langle a_1, \dots, a_k, a_{k+1}, \dots, a_m \rangle$$

In other words, $T(R)$ is the set of prefixes of members of R , ordered by the prefix relation. Clearly, if a tree $\tau = (T(R))$ for some relation R , then this R is unique. We can therefore write R_τ for the unique relation R s.t. $\tau = T(R)$.

In what follows, I use a graphic representation of trees in $\{T(R) : R \in \mathcal{R}^n\}$. For example, let R be the binary relation in (7a). R is graphically represented by the tree in (7b).

(7) a. $R = \{ \langle \text{john}, \text{mary} \rangle, \langle \text{bill}, \text{mary} \rangle, \langle \text{bill}, \text{ruth} \rangle \}$

b.



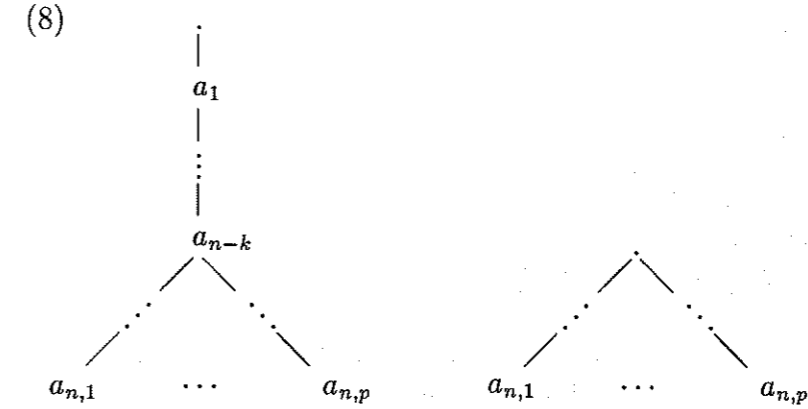
A node α is above a node β and there is a sequence of edges from α to β iff $\alpha, \beta \in N_R$ and $\alpha \preceq \beta$. Every node $\alpha = \langle a_0, \dots, a_k \rangle \in N_R$ is represented by a_k only, while the root node $\langle \rangle$ is represented by a '·'.

Note that the path $\langle \text{john}, \text{mary} \rangle$ in (7b) corresponds to the tuple $\langle \text{john}, \text{mary} \rangle$ in (7a). In general, $\langle a_1, \dots, a_n \rangle$ is a path from the root of a graphic tree representing $T(R)$ to its frontier iff $\langle a_1, \dots, a_n \rangle \in R$ (Lemma 3 in the appendix). So R can be 'read off' any graphic representation of $T(R)$.

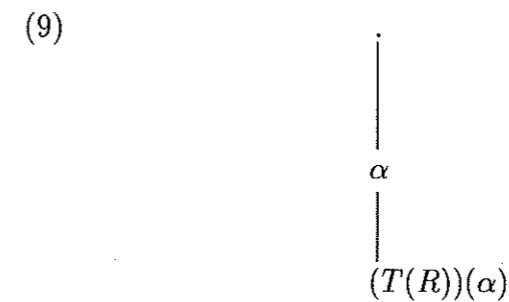
4.3.3 A graphic interpretation of RC

Let R be an n -ary relation, $\alpha = \langle a_1, \dots, a_{n-k} \rangle \in N_R$, and consider the graphic tree representing $T(R)$.

$(T(R))(\alpha)$, or *the tree under α in $T(R)$* , is the graphic object at the right hand side of the drawing in (8).



So the left hand side of (8) can be represented as in (9).



Now consider $R_{(T(R))(\alpha)}$, the k -ary relation determined by $(T(R))(\alpha)$.

$$(10) R_{(T(R))(\alpha)} = \{ \langle a_{(n-k)+1}, \dots, a_n \rangle : \langle a_1, \dots, a_n \rangle \in R \}$$

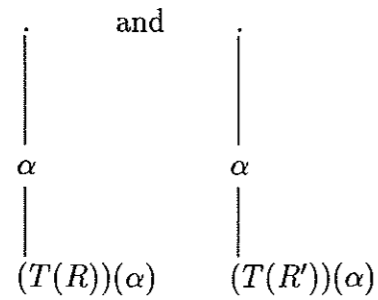
By (1), $\langle a_1, \dots, a_{n-k} \rangle \in g(R)$ iff $g(\{ \langle a_{(n-k)+1}, \dots, a_n \rangle : \langle a_1, \dots, a_n \rangle \in R \}) = 1$. So RC in (5) can be reformulated as in (11).

$$(11) \text{ Let } F \text{ be a GQ of type } \langle n \rangle. F \text{ is type } \langle k \rangle \text{ reducible iff } \exists \text{ a positive GQ } g \text{ of type } \langle k \rangle, k < n, \text{ s.t for all } n\text{-ary relations } R, R', F(R) = F(R') \text{ if for all } \alpha \in \mathcal{E}^{n-k}, g(R_{(T(R))(\alpha)}) = 1 \text{ iff } g(R_{(T(R'))(\alpha)}) = 1.$$

Alternatively, RC can be expressed in terms of Graphic Invariance:

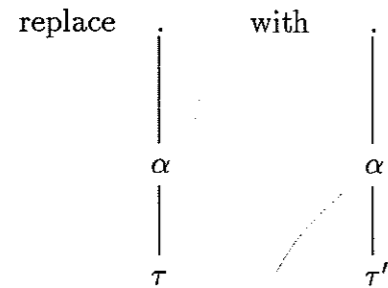
$$(12) \text{ Graphic Invariance}$$

Let F be a GQ of type $\langle n \rangle$. F is type $\langle k \rangle$ reducible iff \exists a positive GQ g of type $\langle k \rangle$, $k < n$, s.t for all n -ary relations R, R' , $F(R) = F(R')$ if for all $\alpha \in \mathcal{E}^{n-k}$,



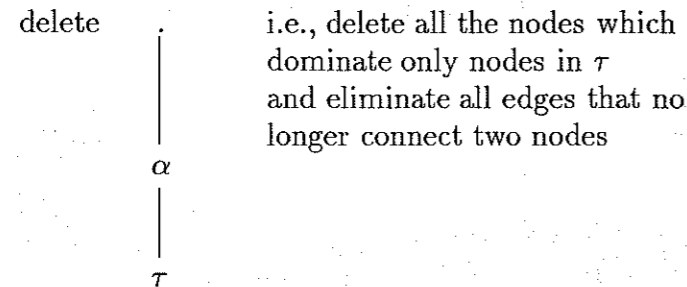
are either identical, or one of them can be derived from the other by one of the following two graphic transformations:

Replace Tree



if $g(R_\tau) = g(R_{\tau'})$, and $R_\tau, R_{\tau'} \neq \emptyset$.

Delete Tree



if $g(R_\tau) = 0 (= g(\emptyset))$.

What this theorem says is that a reducible GQ $F = f \circ g$ is invariant to certain graphic transformations induced by g , i.e., F is *graphically invariant with respect to g* . This fact is the basis of the Graphic Invariance proof technique.

4.3.4 The Graphic Invariance proof technique

Let F be a GQ of type $\langle n \rangle$. To show that F is not type $\langle k \rangle$ reducible we show that there is no positive GQ g of type $\langle k \rangle$ s.t. F is graphically invariant with respect to g .

Concretely, we choose two k -ary relations r and r' and argue as follows:

If $g(r) = 0$, i.e., g is false of r , then F is invariant under deleting a tree representing r . But we exhibit two graphic trees τ_1 and τ_2 that are related by this instance of Delete Tree and $F(R_{\tau_1}) \neq F(R_{\tau_2})$.

Similarly, if $g(r') = 0$ then F is invariant under deleting a tree representing r' . But we exhibit two graphic trees τ_3 and τ_4 that are related by this instance of Delete Tree and $F(R_{\tau_3}) \neq F(R_{\tau_4})$.

The only other option is that $g(r) = g(r') = 1$, in which case F is invariant under replacing a tree representing r by a tree representing r' . But we exhibit two graphic trees τ_5 and τ_6 that are related by this instance of Replace Tree and $F(R_{\tau_5}) \neq F(R_{\tau_6})$.

To make the proofs more perspicuous we write $F(\tau)$ for $F(R_\tau)$.

4.3.5 A sample proof

Let F be the type $\langle 2 \rangle$ GQ determined by the joint interpretation of the subject-object pair in (13).

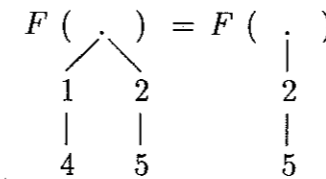
(13) Different people like different things.

Following Keenan (1992) we assume that F is true of LIKE iff there are at least two people, and if x and y are different people, then the things that x likes are not exactly the same as y does. We show F is not type $\langle 1 \rangle$ reducible.

Let $\mathcal{E} = \{1, 2, 3, 4, 5\}$ with 1, 2, 3 being the people in \mathcal{E} and 4, 5 being the things in \mathcal{E} .

We show that there is no positive GQ g of type $\langle 1 \rangle$ such that F is graphically invariant with respect to g .

a. If $g(\{4\}) = 0$ then



by Delete Tree, with

$$\alpha = \langle 1 \rangle \text{ and } (T(R))(\alpha) = \begin{array}{c} \cdot \\ | \\ 4 \end{array}$$

$$\text{But } F \left(\begin{array}{c} \cdot \\ / \quad \backslash \\ 1 \quad 2 \\ | \quad | \\ 4 \quad 5 \end{array} \right) = 1 \neq 0 = F \left(\begin{array}{c} \cdot \\ | \\ 2 \\ | \\ 5 \end{array} \right)$$

b. Similarly if $g(\{5\}) = 0$ then

$$F \left(\begin{array}{c} \cdot \\ / \quad \backslash \\ 1 \quad 2 \\ | \quad | \\ 4 \quad 5 \end{array} \right) = F \left(\begin{array}{c} \cdot \\ | \\ 1 \\ | \\ 4 \end{array} \right)$$

by Delete Tree, with

$$\alpha = \langle 2 \rangle \text{ and } (T(R))(\alpha) = \begin{array}{c} \cdot \\ | \\ 5 \end{array}$$

$$\text{But } F \left(\begin{array}{c} \cdot \\ / \quad \backslash \\ 1 \quad 2 \\ | \quad | \\ 4 \quad 5 \end{array} \right) = 1 \neq 0 = F \left(\begin{array}{c} \cdot \\ | \\ 1 \\ | \\ 4 \end{array} \right)$$

c. Else, $g(\{4\}) = g(\{5\}) = 1$. So

$$F \left(\begin{array}{c} \cdot \\ / \quad \backslash \\ 1 \quad 2 \\ | \quad | \\ 4 \quad 5 \end{array} \right) = F \left(\begin{array}{c} \cdot \\ / \quad \backslash \\ 1 \quad 2 \\ | \quad | \\ 4 \quad 4 \end{array} \right)$$

by Replace Tree, with

$$\alpha = \langle 2 \rangle, (T(R))(\alpha) = \begin{array}{c} \cdot \\ | \\ 5 \end{array}, \text{ and } \tau = \begin{array}{c} \cdot \\ | \\ 4 \end{array}$$

$$\text{But } F \left(\begin{array}{c} \cdot \\ / \quad \backslash \\ 1 \quad 2 \\ | \quad | \\ 4 \quad 5 \end{array} \right) = 1 \neq 0 = F \left(\begin{array}{c} \cdot \\ / \quad \backslash \\ 1 \quad 2 \\ | \quad | \\ 4 \quad 4 \end{array} \right)$$

In the rest of the paper we write proofs like the one above in the following abbreviated format.

$$g(\{4\})=0: \quad F \left(\begin{array}{c} \cdot \\ / \quad \backslash \\ 1 \quad 2 \\ | \quad | \\ 4 \quad 5 \end{array} \right) = F \left(\begin{array}{c} \cdot \\ | \\ 2 \\ | \\ 5 \end{array} \right) \quad \text{Delete Tree}$$

$$g(\{5\})=0: \quad F \left(\begin{array}{c} \cdot \\ / \quad \backslash \\ 1 \quad 2 \\ | \quad | \\ 4 \quad 5 \end{array} \right) = F \left(\begin{array}{c} \cdot \\ | \\ 1 \\ | \\ 4 \end{array} \right) \quad \text{Delete Tree}$$

$$g(\{4\})=g(\{5\})=1: \quad F \left(\begin{array}{c} \cdot \\ / \quad \backslash \\ 1 \quad 2 \\ | \quad | \\ 4 \quad 5 \end{array} \right) = F \left(\begin{array}{c} \cdot \\ / \quad \backslash \\ 1 \quad 2 \\ | \quad | \\ 4 \quad 4 \end{array} \right) \quad \text{Replace Tree}$$

For each two trees, F is true on the relation corresponding to the tree on the left, and false on the relation corresponding to the tree on the right.

NB: unless otherwise noted, F^* is the type $\langle 2 \rangle$ GQ that is the joint interpretation of the subject and object of sentence number (*). For example, $F13$ is the unreducible binary quantifier in the sample proof above.

4.4 Some unreducible GQs

In this section we use the Graphic Invariance proof technique to present some natural language n -ary quantifiers that are not type $\langle k \rangle$ reducible.

Section 4.1 covers all of Keenan's (1992) examples.

Section 4.2 presents related unreducible binary quantifiers whose unreducibility is harder to prove by current proof techniques.

Section 4.3 presents unreducible ternary quantifiers.

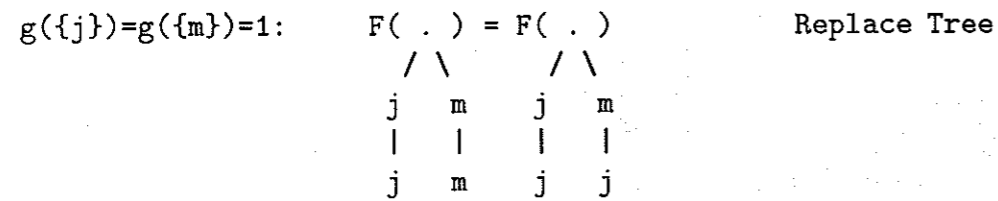
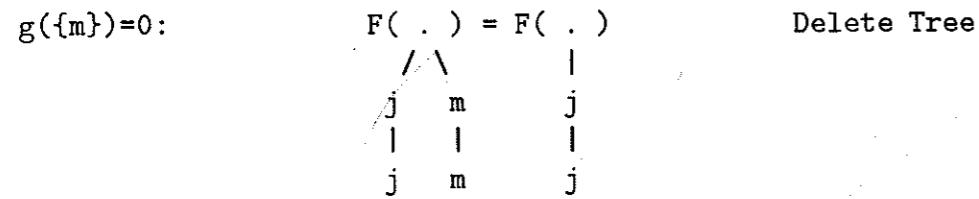
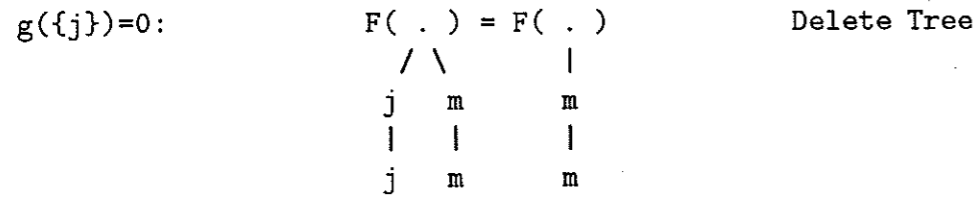
4.4.1

Reflexives:

(14) John and Mary both love themselves.

Let $\mathcal{E} = \{j, m\}$

The proof is identical to the one for *F*13, with 'j' instead of '1' and '4', and 'm' instead of '2' and '5'.

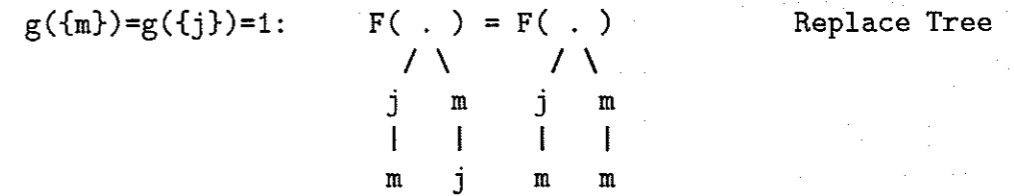
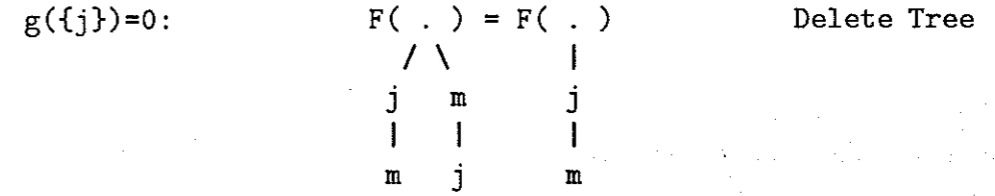
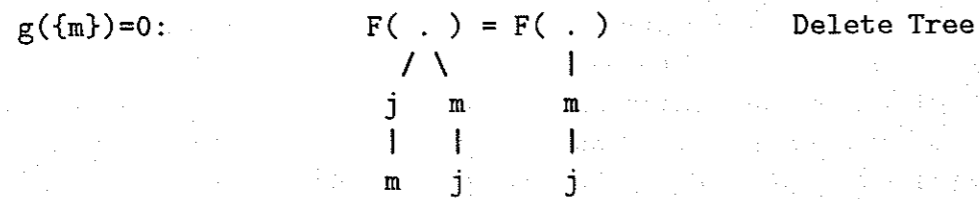


Reciprocals:

(15) John and Mary love each other.

Let $\mathcal{E} = \{j, m\}$

Again, the proof is identical to the one for *F*13, with 'j' instead of '1' and '5', and 'm' instead of '2' and '4'.

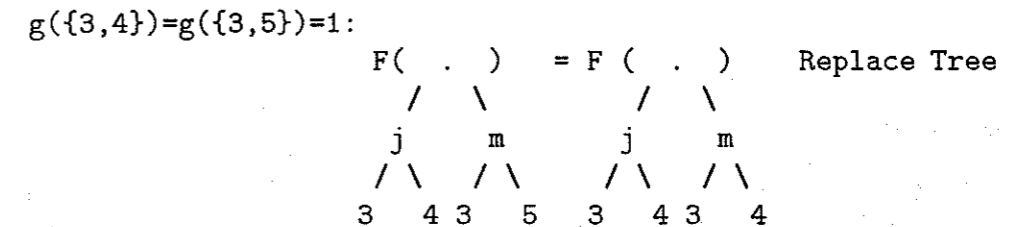
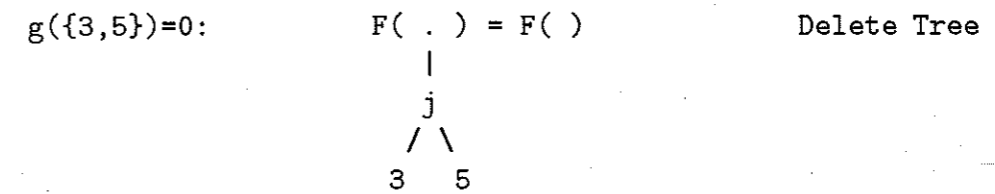
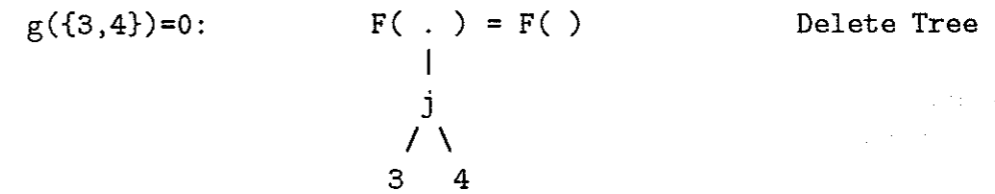


Comparative dependent determiners:

(16) A certain number of professors interviewed a larger number of scholarship applicants.

Let $\mathcal{E} = \{j, m, 3, 4, 5\}$ with j, m being professors and 3, 4, 5 being scholarship applicants.

Let g be a positive GQ of type $\langle 1 \rangle$.



Cumulative quantification:

(17) The editors read a total of two manuscripts between them.

Let $\mathcal{E} = \{1, 2, 4, 5\}$ with 1, 2 being the editors in \mathcal{E} and 4, 5 being manuscripts. The proof is identical to the one for $F13$.

Predicate anaphors:

(18) John knows more students than Mary (does).

Let $\mathcal{E} = \{j, m, 3, 4, 5\}$ with 3, 4 being the students in \mathcal{E} . The proof is identical to the one for $F16$.

'Else-else' anaphora:

(19) John criticized Bill and no one else criticized anyone else.

Let $\mathcal{E} = \{j, b\}$.

Let g be a positive GQ of type $\langle 1 \rangle$.

$g(\{b\})=0:$ $F(\cdot) = F(\cdot)$ Delete Tree
 $\begin{array}{c} | \\ j \\ | \\ b \end{array}$

$g(\{j\})=0:$ $F(\cdot) = F(\cdot)$ Delete Tree
 $\begin{array}{c} | \quad / \quad \backslash \\ j \quad j \quad b \\ | \quad | \quad | \\ b \quad b \quad j \end{array}$

$g(\{b\})=g(\{j\})=1:$ $F(\cdot) = F(\cdot)$ Replace Tree
 $\begin{array}{c} | \quad | \\ j \quad j \\ | \quad | \\ b \quad j \end{array}$

Branching quantifiers:

Let $F20$ be the branching quantifier determined by AT LEAST TWO, defined as in (20). $F20$ is a possible interpretation of the subject-object pair in the sentence *At least two kids climbed at least two trees*.

(20) $(\text{AT LEAST TWO})(A) * (\text{AT LEAST TWO})(B)(R) = 1$ iff there are $A' \subseteq A, B' \subseteq B$ s.t. $|A'| > 1$ and $|B'| > 1$ and $A' \times B' \subseteq R$.

Let $\mathcal{E} = \{1, 2, 3, 4, 5\}$ with $A = \{1, 2\}$ and $B = \{3, 4, 5\}$.

Let g be a positive GQ of type $\langle 1 \rangle$.

$g(\{3, 4\})=0:$ $F(\cdot) = F(\cdot)$ Delete Tree
 $\begin{array}{c} / \quad \backslash \quad \quad | \\ 1 \quad 2 \quad \quad 2 \\ / \quad \backslash \quad / \quad \backslash \quad \quad / \quad \backslash \\ 3 \quad 4 \quad 3 \quad 4 \quad \quad 3 \quad 4 \end{array}$

$g(\{3, 5\})=0:$ $F(\cdot) = F(\cdot)$ Delete Tree
 $\begin{array}{c} / \quad \backslash \quad \quad | \\ 1 \quad 2 \quad \quad 1 \\ / \quad \backslash \quad / \quad \backslash \quad \quad / \quad \backslash \\ 3 \quad 5 \quad 3 \quad 5 \quad \quad 3 \quad 5 \end{array}$

$g(\{3, 4\})=g(\{3, 5\})=1:$ $F(\cdot) = F(\cdot)$ Replace Tree
 $\begin{array}{c} / \quad \backslash \quad \quad / \quad \backslash \\ 1 \quad 2 \quad \quad 1 \quad 2 \\ / \quad \backslash \quad / \quad \backslash \quad / \quad \backslash \\ 3 \quad 5 \quad 3 \quad 5 \quad 3 \quad 5 \quad 3 \quad 4 \end{array}$

Bach-Peters sentences (Higginbotham and May (1981)):

A typical Bach-Peters sentence is *Every pilot that shot at it hit some mig that chased him*.

For every binary relation R let $F21$ be the Bach-Peters-related type $\langle 2 \rangle$ GQ defined by $(\text{EVERY} * \text{SOME})(R)$ in (21).

(21) $(\text{EVERY} * \text{SOME})(R)(S) = 1$ iff for every a s.t. $\{b : \langle a, b \rangle \in R\}$, there is a c s.t. $\langle a, c \rangle \in R$ and $\langle a, c \rangle \in S$.

This is a complicated GQ, but the Graphic Invariance proof that it is unreducible is easy. Let $\mathcal{E} = \{1, 2, 4, 5\}$, $R = \{\langle 1, 4 \rangle, \langle 2, 5 \rangle\}$. The proof is identical to the one for $F13$.

4.4.2 New binary unreducible GQs

Keenan's examples contain a curious gap. For example, they include the distributive readings of sentences like (22)-(24), but not of sentences like (25).

- (22) Every student criticized himself.
- (23) John and Bill criticized themselves.
- (24) The two students criticized themselves.
- (25) Two students criticized themselves.

This gap is not accidental, since Keenan proves the unreducibility of the quantifiers in (22)-(24) by the theorem in (26).

- (26) Reducibility Equivalence (RE) For F, G reducible GQs of type $\langle 2 \rangle$,
 $F = G$ iff for all subsets P, Q of \mathcal{E} , $F(P \times Q) = G(P \times Q)$

A typical argument is the following: F_{22} has the same values on the cross product relations $P \times Q$ as F_{27} , but differs from it on the diagonal $\{ \langle a, a \rangle : a \in \mathcal{E}, \text{ and } a \text{ is a student} \}$, if \mathcal{E} contains more than one student. $F_{27} = (\text{EVERY STUDENT}) \circ (\text{EVERY STUDENT})$ and is therefore reducible, so F_{22} is unreducible by RE.

- (27) Every student criticized every student.

This type of argument does not go through for GQs like F_{28} , which is determined by an indefinite subject NP: Restricted to the cross products relations $P \times Q$, F_{28} is identical to F_{29} , rather to F_{30} . (Distributive readings)

- (28) Two students criticized themselves.
- (29) Two students criticized the same two students.
- (30) Two students criticized two students.

For example, F_{28} can be distinguished from F_{30} on the universe $\mathcal{E} = \{1, 2, 3, 4\}$ where 1, 2, 3, 4 are students. F_{28} is false on $\{1, 2\} \times \{3, 4\}$ but F_{30} is true. F_{28} is identical to F_{29} , but as will be prove shortly, F_{29} itself is unreducible. So RE cannot be applied as in the examples above.

This is a special case of a more general problem, which arises if the subject and object are related by other relations than 'self' too. For example, consider the 'mother' relation in (31) and (32). Restricted to the cross product relations $P \times Q$ F_{31} is identical to F_{32} , which is itself unreducible.

- (31) A husband and wife kissed their (own) mothers.
- (32) A husband and wife kissed their (own) mothers and mothers-in-law.

Using Graphic Invariance, it is easy to prove these GQs are indeed unreducible.

- (33) A husband and wife each kissed their (own) mother and mother-in-law.

Let $\mathcal{E} = \{1, 2, 3, 4, m1, m2, m3, m4\}$ such that 1 and 2 are a man and his wife, 3 and 4 are a man and his wife, $m1$ is the mother of 1, $m2$ is the mother of 2, $m3$ is the mother of 3 and $m4$ is the mother of 4.

Let g be a positive GQ of type $\langle 1 \rangle$.

The structure of the proof is identical to the one for F_{20} .

$$g(\{m1, m2\})=0: \quad \begin{array}{c} F(\quad . \quad) = F(\quad . \quad) \quad \text{Delete Tree} \\ \begin{array}{c} / \quad \backslash \\ 1 \quad 2 \\ / \quad \backslash \quad / \quad \backslash \\ m1 \quad m2 \quad m1 \quad m2 \end{array} \quad \begin{array}{c} | \\ 2 \\ / \quad \backslash \\ m1 \quad m2 \end{array} \end{array}$$

$$g(\{m3, m4\})=0: \quad \begin{array}{c} F(\quad . \quad) = F(\quad . \quad) \quad \text{Delete Tree} \\ \begin{array}{c} / \quad \backslash \\ 3 \quad 4 \\ / \quad \backslash \quad / \quad \backslash \\ m3 \quad m4 \quad m3 \quad m4 \end{array} \quad \begin{array}{c} | \\ 3 \\ / \quad \backslash \\ m3 \quad m4 \end{array} \end{array}$$

$$g(\{m1, m2\})=g(\{m3, m4\})=1: \quad \begin{array}{c} F(\quad . \quad) = F(\quad . \quad) \quad \text{Replace Tree} \\ \begin{array}{c} / \quad \backslash \\ 1 \quad 2 \\ / \quad \backslash \quad / \quad \backslash \\ m1 \quad m2 \quad m1 \quad m2 \end{array} \quad \begin{array}{c} / \quad \backslash \\ 1 \quad 2 \\ / \quad \backslash \quad / \quad \backslash \\ m1 \quad m2 \quad m3 \quad m4 \end{array} \end{array}$$

- (34) A husband and wife kissed their (own) mother.

Assuming the same \mathcal{E} as for F_{33} , the proof is identical to the one for F_{33} .

- (35) Two students criticized themselves.
- (36) Two students criticized the same two students.

Let $\mathcal{E} = \{1, 2, 3, 4\}$ such that 1, 2, 3, 4 are students.

The proofs are again identical to the one for F_{33} , with '1' instead of 'm1', '2' instead of 'm2', '3' instead of 'm3' and '4' instead of 'm4'.

$$g(\{1,2\})=0 : \quad \begin{array}{c} F(\quad) = F(\quad) \quad \text{Delete Tree} \\ / \quad \backslash \quad \quad | \\ 1 \quad 2 \quad \quad 2 \\ / \quad \backslash \quad / \quad \backslash \\ 1 \quad 2 \quad 1 \quad 2 \quad 1 \quad 2 \end{array}$$

$$g(\{3,4\})=0 : \quad \begin{array}{c} F(\quad) = F(\quad) \quad \text{Delete Tree} \\ / \quad \backslash \quad \quad | \\ 3 \quad 4 \quad \quad 3 \\ / \quad \backslash \quad / \quad \backslash \\ 3 \quad 4 \quad 3 \quad 4 \quad 3 \quad 4 \end{array}$$

$$g(\{1,2\})=g(\{3,4\})=1 : \quad \begin{array}{c} F(\quad) = F(\quad) \quad \text{Replace Tree} \\ / \quad \backslash \quad \quad / \quad \backslash \\ 1 \quad 2 \quad \quad 1 \quad 2 \\ / \quad \backslash \quad / \quad \backslash \\ 1 \quad 2 \quad 1 \quad 2 \quad 1 \quad 2 \quad 3 \quad 4 \end{array}$$

4.4.3 Unreducible ternary GQs

The current literature on unreducible quantifiers deals exclusively with binary quantifiers. Binary quantifiers are either reducible, i.e., type $\langle 1 \rangle$ reducible, or unreducible, i.e., not type $\langle 1 \rangle$ reducible.

Ternary quantifiers can be completely reducible, i.e., both type $\langle 1 \rangle$ and type $\langle 2 \rangle$ reducible. For example, the ternary quantifier determined by the sentence *Most department heads introduced at least one student to two professors* can be expressed as MOST DEPARTMENT HEADS \circ AT LEAST ONE STUDENT \circ TWO PROFESSORS. It is therefore both type $\langle 1 \rangle$ and type $\langle 2 \rangle$ reducible.

Ternary quantifiers can also be completely unreducible, i.e., neither type $\langle 1 \rangle$ nor type $\langle 2 \rangle$ reducible. For example, the following sentences determine completely unreducible quantifiers: *Different teachers assigned different questions to different students*, *Two guests introduced themselves to each other* and *John gave his girlfriend more presents than Bill did*.

But in addition ternary quantifiers can be type $\langle 1 \rangle$ reducible but not type $\langle 2 \rangle$ reducible, or type $\langle 2 \rangle$ reducible but not type $\langle 1 \rangle$ reducible.

For example, the ternary quantifier determined by the sentence *Mary introduced each guest to his host* can be expressed as MARY \circ EACH GUEST_HIS HOST, where EACH GUEST STUDENT_HIS HOST is the unreducible binary quantifier determined by the sentence *Each guest greeted his host*. This ternary quantifier is type $\langle 1 \rangle$ but not type $\langle 2 \rangle$ reducible.

Similarly, the ternary quantifier determined by the sentence *Each guest introduced his host to Mary* can be expressed as EACH GUEST_HIS HOST \circ MARY. This ternary quantifier is type $\langle 2 \rangle$ but not type $\langle 1 \rangle$ reducible.

In the rest of this section, we prove the unreducibility claims made about the ternary quantifiers above.

Let $F37$ be the type $\langle 3 \rangle$ GQ that is the joint interpretation of the subject-direct object-indirect object triplet in (37).

(37) Different teachers assigned different questions to different students.

We assume that (37) is true of ASSIGN iff there are at least two teachers, and every two different teachers did not assign exactly the same questions to exactly the same students. In other words, for all distinct teachers x and y we can find some question-student pair (q, s) s.t. x assigned q to s and y did not, or vice versa.

We show that there are no GQs f, g , g positive, such that $F37 = f \circ g$ and either f is of type $\langle 2 \rangle$ and g is of type $\langle 1 \rangle$, or f is of type $\langle 1 \rangle$ and g is of type $\langle 2 \rangle$.

Let $\mathcal{E} = \{1, 2, 3, 4, 5, 6, 7\}$ with 1, 2, 3 being the teachers in \mathcal{E} , 4, 5 being the students in \mathcal{E} , and 6, 7 being the questions in \mathcal{E} .

Let g be a positive GQ of type $\langle 1 \rangle$.

$$g(\{6\})=0 : \quad \begin{array}{c} F(\quad) = F(\quad) \quad \text{Delete Tree} \\ / \quad \backslash \quad \quad | \\ 1 \quad 2 \quad \quad 2 \\ | \quad | \quad \quad | \\ 4 \quad 4 \quad \quad 4 \\ | \quad | \quad \quad | \\ 6 \quad 7 \quad \quad 7 \end{array}$$

$$g(\{7\})=0 : \quad \begin{array}{c} F(\quad) = F(\quad) \quad \text{Delete Tree} \\ / \quad \backslash \quad \quad | \\ 1 \quad 2 \quad \quad 1 \\ | \quad | \quad \quad | \\ 4 \quad 4 \quad \quad 4 \\ | \quad | \quad \quad | \\ 6 \quad 7 \quad \quad 6 \end{array}$$

$g(\{6\})=g(\{7\})=1:$ $F(\cdot) = F(\cdot)$

```

  / \   / \
 1  2  1  2
 |  |  |  |
 4  4  4  4
 |  |  |  |
 6  7  6  6

```

Replace Tree

Let g be a positive GQ of type $\langle 2 \rangle$.

$g(\{<4,6>\})=0:$ $F(\cdot) = F(\cdot)$

```

  / \   |
 1  2   2
 |  |   |
 4  4   4
 |  |   |
 6  7   7

```

Delete Tree

$g(\{<4,7>\})=0:$ $F(\cdot) = F(\cdot)$

```

  / \   |
 1  2   1
 |  |   |
 4  4   4
 |  |   |
 6  7   6

```

Delete Tree

$g(\{<4,6>\})=$
 $g(\{<4,7>\})=1:$ $F(\cdot) = F(\cdot)$

```

  / \   / \
 1  2  1  2
 |  |  |  |
 4  4  4  4
 |  |  |  |
 6  7  6  6

```

Replace Tree

(38) Two guests introduced themselves to each other.

Interpreted so that (38) is true of INTRODUCE iff there are two distinct guests x, y such that x introduced x to y , and y introduced y to x .

Let $\mathcal{E} = \{1, 2\}$ with 1, 2 being guests in \mathcal{E} .

Let g be a positive GQ of type $\langle 1 \rangle$.

$g(\{2\})=0:$ $F(\cdot) = F(\cdot)$

```

  / \   |
 1  2   2
 |  |   |
 1  2   2
 |  |   |
 2  1   1

```

Delete Tree

$g(\{1\})=0:$ $F(\cdot) = F(\cdot)$

```

  / \   |
 1  2   1
 |  |   |
 1  2   1
 |  |   |
 2  1   2

```

Delete Tree

$g(\{2\})=g(\{1\})=1:$ $F(\cdot) = F(\cdot)$

```

  / \   / \
 1  2  1  2
 |  |  |  |
 1  2  1  2
 |  |  |  |
 2  1  2  2

```

Replace Tree

Let g be a positive GQ of type $\langle 2 \rangle$.

$g(\{<1,2>\})=0:$ $F(\cdot) = F(\cdot)$

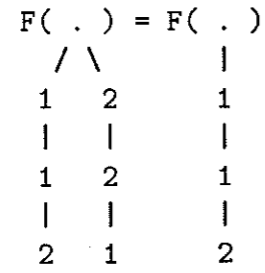
```

  / \   |
 1  2   2
 |  |   |
 1  2   2
 |  |   |
 2  1   1

```

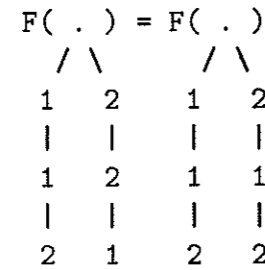
Delete Tree

$g(\{<2,1>\})=0:$



Delete Tree

$g(\{<1,2>\})=$
 $g(\{<2,1>\})=1:$



Replace Tree

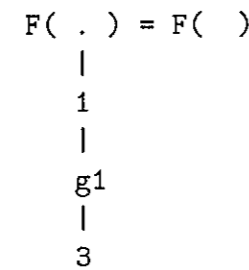
(39) John gave his girlfriend more presents than Bill did.

We interpret (39) as true of GIVE iff the number of presents that John gave to John's girlfriend is larger than the number of presents that Bill gave to Bill's girlfriend.

Let $\mathcal{E} = \{1, 2, g1, g2, 3, 4\}$ with 1 being John, 2 being Bill, $g1$ being John's girlfriend, $g2$ being Bill's girlfriend and 3, 4 being a present in \mathcal{E} .

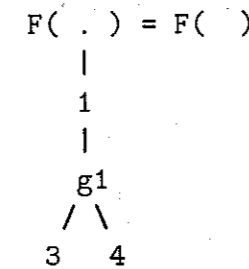
Let g be a positive GQ of type $\langle 1 \rangle$.

$g(\{3\})=0:$



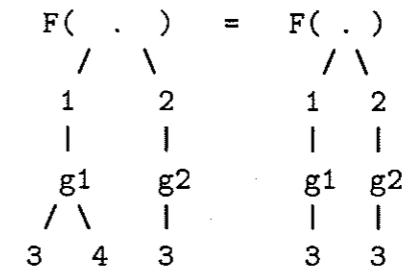
Delete Tree

$g(\{3,4\})=0:$



Delete Tree

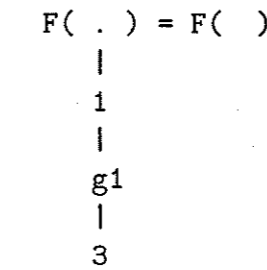
$g(\{3\})=g(\{3,4\})=1:$



Replace Tree

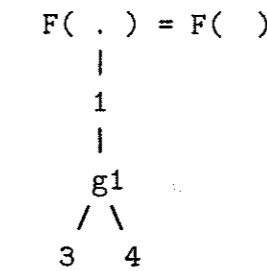
Let g be a positive GQ of type $\langle 2 \rangle$.

$g(\{<g1,3>\})=0:$



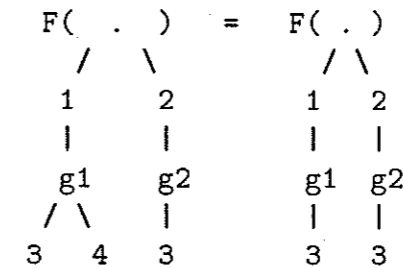
Delete Tree

$g(\{<g1,3>, <g1,4>\})=0:$



Delete Tree

$g(\{<g1,3>\})=$
 $g(\{<g1,3>, <g1,4>\})=1:$



Replace Tree

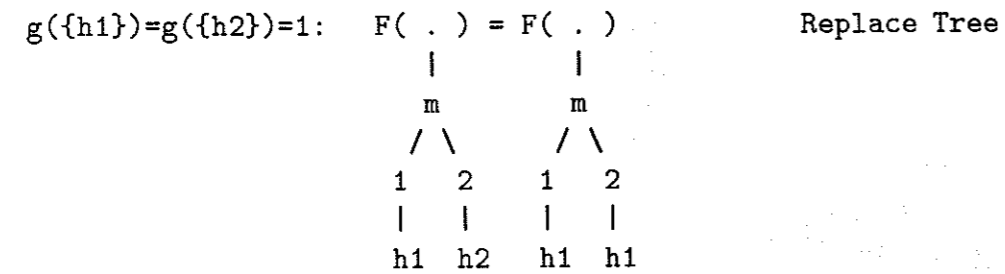
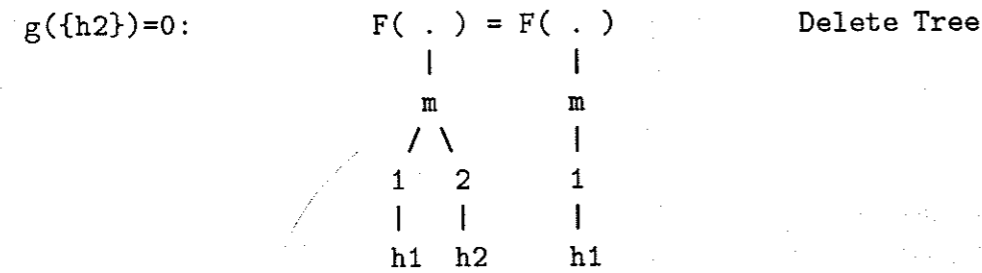
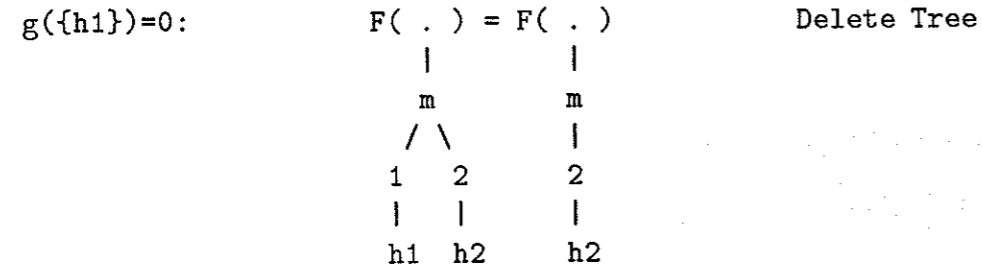
(40) Mary introduced each guest to his host.

(41) Each guest introduced his host to Mary.

Let $\mathcal{E} = \{m, 1, 2, h1, h2\}$ with 1 and 2 being the guests in \mathcal{E} , $h1$ being 1's host and $h2$ being 2's host.

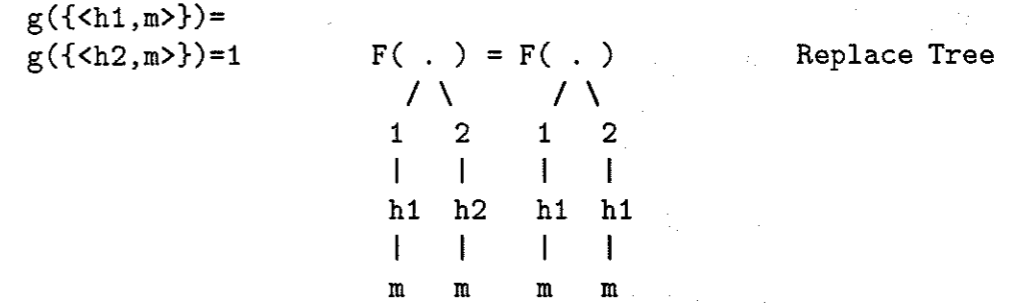
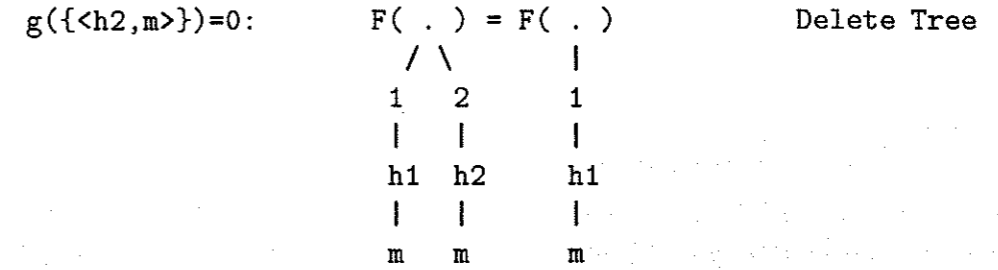
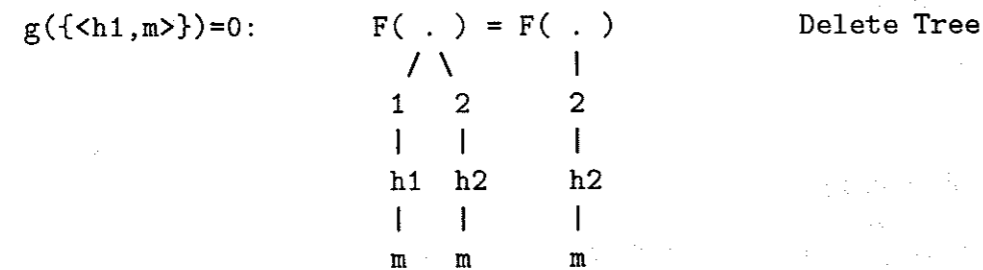
We show that there are no GQ f, g , g positive, such that $F40 = f \circ g$ and f is of type $\langle 2 \rangle$ and g is of type $\langle 1 \rangle$.

Let g be a positive GQ of type $\langle 1 \rangle$.



We show that there are no GQ f, g , g positive, such that $F41 = f \circ g$ and f is of type $\langle 1 \rangle$ and g is of type $\langle 2 \rangle$.

Let g be a positive GQ of type $\langle 2 \rangle$.



Appendix

Let $R \in \mathcal{R}^n$. $T(R)$, the tree determined by R , is the set of prefixes of members of R , ordered by the prefix relation.

Let $\alpha = \langle a_1, \dots, a_n \rangle$.

Lemma 1 $\alpha \in N_R$ iff $\alpha \in R$.

Suppose $\alpha \in N_R$.

Let $Path(\alpha) = \langle \beta_1, \dots, \beta_n \rangle$, where $\beta_i = \langle a_1, \dots, a_i \rangle$.

Let $Label(\alpha)$ be the last element of α .

Define $Label(Path(\alpha)) = \langle Label(\beta_1), \dots, Label(\beta_n) \rangle$.

Lemma 2 $Label(Path(\alpha)) = \langle a_1, \dots, a_n \rangle = \alpha$.

In a graphic tree representing $T(R)$, $Label(Path(\alpha))$ is a sequence of nodes along the edges from the root to the frontier.

Lemma 3 $\alpha \in R$ iff $\alpha \in N_R$ and $Label(Path(\alpha)) = \alpha$.

In a graphic tree representing $T(R)$, R can be 'read off' the nodes along the edges from the root to the frontier.

References

- van Benthem, Johan. 1989. Polyadic Quantifiers. *Linguistics and Philosophy* 12, 437 – 465.
- Higginbotham, James and Robert May. 1981. Questions, Quantifiers and Crossing. *The Linguistic Review* 1, 41 – 79.
- Keenan, Edward L. 1987. Unreducible n-ary Quantifiers in Natural Language. In Peter Gärdenfors, ed., *Generalized Quantifiers*, 109 – 150. Dordrecht: Reidel.
- Keenan, Edward L. 1992. Beyond the Frege Boundary. *Linguistics and Philosophy* 15, 199 – 221.

Chapter 5

Conservativity and Extension*

5.1 Introduction

In general, a generalized quantifier is a functional that assigns to every non-empty model M a binary relation between subsets of M . In other words, given a model M and two of its subsets, it says 'yes' or 'no'. But the generalized quantifiers used in natural language semantics seem to obey additional conditions.

In particular, the following two have been claimed to be universally true: *conservativity* says that $Q_M AB = Q_M AA \cap B$; *extension* says that if A and B are subsets of both M and M' , $Q_M AB = Q_{M'} AB$. Taken together, these two conditions express *domain restriction* (van Benthem (1984)), a basic asymmetry between the roles of the two subsets taken by the generalized quantifier. One subset serves to define the domain of quantification, while the other is an extension of some property in that restricted domain. For example, it is standardly assumed that the semantics of the sentence *Every woman laughs* involves the quantification structure $\forall_M WL$, where W is the set of entities in M who are woman and L is the set of entities in M who laugh. Domain restriction then says that the value of this sentence in M depends only on entities in M that are in W . Thought about in terms of domain restriction, conservativity and extension lead one to think of modal logic, which is a natural system for expressing restricted quantification: the value of a formula of the form $\Box\phi$ at a point s in a model $\mathcal{M} = (S, R, V)$ depends only on points in \mathcal{M} that are successors of s in the accessibility relation R .

This chapter suggests a connection between generalized quantifiers and modal operators, and uses it to show that conservativity and extension together correspond to a basic modal invariance, called invariance under *generated submodels*: the value of a formula ψ at a point s in a model \mathcal{M} depends only on points in \mathcal{M} that are accessible from s by a finite number of R steps.

*I would like to thank Johan van Benthem, Jan van Eijck, and Andras Simon for their helpful comments on this subject.

5.2 Generalized quantifiers and modal operators

Definition 1 A generalized quantifier is a functional Q which to each non-empty set M assigns a binary relation Q_M between subsets of M .

A generalized quantifier Q can be thought of as inducing a unary model operator \Box_Q whose semantics is defined by:

Definition 2

$$\mathcal{M}, s \models \Box_Q \phi \text{ iff } Q_W \{s' \mid sRs'\} \{s' \mid \mathcal{M}, s' \models \phi\}$$

For example, the standard \Box and \Diamond are equal to \Box_{\forall} and \Box_{\exists} , respectively.

5.3 Conservativity and extension

Definition 3 A generalized quantifier Q is said to have CONSERV if the following condition holds for any non-empty sets M, M' and $A, B \subseteq M, A', B' \subseteq M'$.

$$Q_M AB \Leftrightarrow Q_M AA \cap B$$

Similarly, it is said to have EXT if the following condition holds

$$\text{if } A, B \subseteq M \subseteq M' \text{ then } Q_M AB \Leftrightarrow Q_{M'} AB$$

Definition 4 \mathcal{M}^s , the submodel of \mathcal{M} generated by s , is defined by:

$$\mathcal{M}^s =_{def} \langle W^s, R^s, V^s \rangle, \text{ where}$$

$$W^s = \{s' \mid sR^i s', 0 \leq i\},$$

$$R^s = R \cap (W^s \times W^s),$$

$$V^s(p) = V(p) \cap W^s$$

In other words, the submodel of \mathcal{M} generated by s is \mathcal{M} restricted to the points reachable from S by a finite number of R steps.

Theorem $\Box_Q p$ is invariant under generated submodels iff Q has CONSERV and EXT.

proof:

\Leftarrow

Suppose Q has CONSERV and EXT. Let M and M' be non-empty sets, $A, B \subseteq M, A, C \subseteq M', M \subseteq M'$, and $A \cap B = A \cap C$. Then $Q_M AB$ iff $Q_M AA \cap B$ (by CONSERV) iff $Q_{M'} AA \cap C$ (by EXT) iff $Q_{M'} AC$ (by CONSERV). Let $\mathcal{M}^s = \langle W^s, R^s, V^s \rangle$ be the submodel of $\mathcal{M} = \langle W, R, V \rangle$ generated by $s \in W$. $\{t \mid sR^s t\} = \{t \mid sRt\}$, and $\{t \mid sR^s t\} \cap \{t \mid \mathcal{M}^s, t \models p\} = \{t \mid sRt\} \cap \{t \mid \mathcal{M}, t \models p\}$. So $\mathcal{M}, s \models \Box_Q p$ iff $\mathcal{M}^s, s \models \Box_Q p$ by the truth definition of $\Box_Q p$.

\Rightarrow

Suppose $\Box_Q p$ is invariant under generated submodels. Let $A, B \subseteq M \subseteq M'$. For any non empty set M and $A, B \subseteq M$ let $\mathcal{M}_{(M,A,B)}$, the Kripke model determined by (M, A, B) , be defined by:

$$\mathcal{M}_{(M,A,B)} =_{def} \langle M, M \times A, B \rangle$$

In other words, the domain of \mathcal{M} is M , the set of R -successors for every point in M are the points in A , and for every propositional letter $p, V(p) = B$. Let $A, B \subseteq M \subseteq M'$, and s be an element of M , and consider the Kripke models $\mathcal{M}_{(M,A,B)}$ and $\mathcal{M}_{(M',A,B)}$. By the construction of $\mathcal{M}_{(M,A,B)}$, $Q_M AB$ iff $\mathcal{M}_{(M,A,B)}, s \models \Box_Q p$. Similarly, $Q_{M'} AB$ iff $\mathcal{M}_{(M',A,B)}, s \models \Box_Q p$. Since $\Box_Q p$ is invariant under generated submodels, $\mathcal{M}_{(M,A,B)}, s \models \Box_Q p$ iff $\mathcal{M}_{(M,A,B)}^s, s \models \Box_Q p$ and $\mathcal{M}_{(M',A,B)}, s \models \Box_Q p$ iff $\mathcal{M}_{(M',A,B)}^s, s \models \Box_Q p$. But $\mathcal{M}_{(M,A,B)}^s$ is identical to $\mathcal{M}_{(M',A,B)}^s$. So $Q_M AB$ iff $Q_{M'} AB$, i.e., Q has EXT. Next let $A, B \subseteq M$, and let s be an element not in M . Since Q has EXT, $Q_M AB$ iff $Q_{M \cup \{s\}} AB$. Similarly, $Q_M AA \cap B$ iff $Q_{M \cup \{s\}} AA \cap B$. Since $\mathcal{M}_{(M \cup \{s\}, A, B)}^s$ is identical to $\mathcal{M}_{(M \cup \{s\}, A, A \cap B)}^s$, $Q_{M \cup \{s\}} AB$ iff $Q_{M \cup \{s\}} AA \cap B$ by the same argument as for EXT. So Q has CONSERV. \square

5.4 Conclusion

This chapter suggests a connection between generalized quantifiers and modal operators and uses it to show that the natural language universals conservativity and extension correspond to the basic modal invariance under generated submodels.

Acknowledgements

I would like to acknowledge the financial support of the Netherlands Organization for Scientific Research (NWO), project NF 102/62-356 'Structural and Semantic Parallels between Natural Languages and Programming Languages'.

References

Bentham, J.v.: 1984, Questions about Quantifiers, *Journal of Symbolic Logic* 49, 443 - 466.