A STUDY OF NATURAL LANGUAGE QUANTIFICATION AND ANAPHORA THROUGH FAMILIES OF SETS AND BINARY RELATIONS

by

Robert Lizée

A thesis submitted in conformity with the requirements
for the degree of Master of Science,
Graduate Department of Computer Science,
University of Toronto

one two

five

three four

A STUDY OF NATURAL LANGUAGE QUANTIFICATION AND ANAPHORA THROUGH FAMILIES OF SETS AND BINARY RELATIONS

Master of Science 1995

Robert Lizée

Department of Computer Science

University of Toronto

Abstract

In this thesis, we study the use of families of sets and binary relations to represent natural language quantification and anaphora. We focus on quantification in a dependency-free context (no variables) using the idea of Barwise and Cooper (1981) of expressing quantifiers as families of sets. A language where sentences are expressed as subset relations between generalized quantifiers is shown equivalent to the variable-free Montagovian syntax of McAllester and Givan (1992), relating their notion of obvious inference to the transitive closure. To account for anaphora, we propose to use an extended algebra of binary relations (Suppes, 1976; Böttner, 1992), in practice, restricting the number of variables to one. We contribute to the formalism with a family of composition operators, allowing to account for sentences with determiners other than *'every'*, *'some'*, or *'no'*. Moreover, we show how to handle some cases of long-distance anaphora, some cases involving the word *'other'*, and some donkey-sentence anaphora.

Acknowledgments

I wish to thank my supervisor Graeme Hirst for his help. I also benefited from the comments of Jeff Siskind. Finally, special thanks to my second reader Joachim von zur Gathen.

This work was funded by the Natural Sciences and Engineering Research Council of Canada.

Contents

Chapter 1

Introduction

In this thesis, we study the use of families of sets and binary relations to represent natural language quantification and anaphora.

One belief that underlies this work is that standard first-order logic notation is inadequate for natural language because of structural differences, for instance, with regards to quantification and anaphora. The first-order quantifiers ($\forall$ and $\exists$) don't distinguish between the domain of quantification and the quantified predicate; for example, to represent sentence (1.1), the denotation of *'man'* and *'bald'* must be merged into a single predicate ($\texttt{man}(x) \rightarrow \texttt{bald}(x)$) to be quantified by $\forall$. The notion of anaphora in natural language does not directly match the use of variables in first-order logic. First, it would be surprising that an arbitrary number of variables is required to represent natural language sentences. Secondly, the use of variables alone cannot explain every type of anaphora while preserving its structure. For example, the donkey sentence (1.2) cannot be represented by simply replacing *'it'* by a variable bound to *'a donkey'*, since *'it'* is not in the scope of the existential quantifier induced by *'a donkey'*. To obtain its first-order representation, the sentence must be reformulated in something like sentence (1.3). (The portion in parentheses accounts for the presupposition that *'if a man owns a donkey, he owns only one donkey'*). Note that these deficiencies are not unique to first-order logic notation, since many other logics and extensions of first-order logic use first-order quantifiers and variables in a similar way.

(1.1) Every man is bald. $\forall x,\ \texttt{man}(x) \rightarrow \texttt{bald}(x)$

(1.2) Every man who owns a donkey beats it.

$$\forall x,\ [\texttt{man}(x) \wedge \exists y,\ (\texttt{own}(x,y) \wedge \texttt{donkey}(y))] \rightarrow \texttt{beat}(x,y?)$$

(1.3) Every man (owns $\rightarrow$ beats) every donkey (and every man owns at most one donkey) .

There are many reasons why a formalism close to natural language is desirable. First, it facilitates the conversion of English utterances into semantic representations and vice-versa. But also, it is conceivable that the shape of natural language—its constraints, its syntax, etc...—reflects inner representations people use to think. For instance, some syntactic constraints observed might be in fact semantic. Consequently, the search for a representation mimicking natural language may be revealing with regard to semantic issues such as what people infer easily and how people deal with ambiguous sentences.

In this thesis, we are committed to study quantification and anaphora from a non-traditional perspective. Apart from the justifications given above, valuable insights leading to a better understanding of quantification and anaphora in natural language might be gained from exploring alternative representations. The thesis is divided into two main parts, one focusing on quantification and one focusing on anaphora. We conclude in a third part by unifying the representation of the two previous parts. A forth part contains the appendices.

In part I, we focus on quantification in a dependency-free context (no variables). In chapter 2, we present the theory of generalized quantifiers proposed by Barwise and Cooper (1981) for the study of quantification in natural language. The appeal of this theory is due to its conformity to the notion that each syntactic unit in natural language has a semantic counterpart (Montague, 1974). In particular, its prevalent semantic units are the quantifiers, defined as families of sets, whose natural language counterparts are the simple NP's (simple noun phrases), like *'a man'*, *'most men'*, and *'every man'*. Our contribution is the observation that some simple English sentences are expressible as a subset relation between such families of sets. The advantage of this representation is that new sentences are inferable by performing the transitive closure. In fact, in chapter 3, we show that the variable-free Montagovian syntax of McAllester and Givan (1992) is equivalent to a simple logic based on subset relations between generalized quantifiers and using to a set of inference rules built around the transitivity rule. The interest of the variable-free Montagovian syntax lies in the notion of obviousness attached to it—which might serve as a basis to study human inferences, since it is based (at least crudely) on the syntax of natural language. Our new version of the logic relates that notion of obviousness to the transitivity rule, since it is its most complex rule.

In part II, we focus on anaphora. The goal is to enlarge the fragment of English captured by the Montagovian syntax with anaphora—the long-term goal, not covered in this thesis, is an extension of the notion of obviousness to include anaphora. Instead of offering an arbitrary number of variables, we propose to use an extended algebra of binary relations (Suppes, 1976; Böttner, 1992), in practice, restricting the number of variables to one. We contribute to the formalism with a family of composition operators, allowing the account of sentences with quantifiers other than *'every'*, *'some'*, or *'no'*. Moreover, we show how to handle some cases of long distance anaphora, some cases involving the word *'other'*, and some donkey sentences. Actually, anaphora in the donkey sentence (1.2) is resolved by handling *'it'* as a function of x returning *'the donkey owned by x'*, reducing the problem to the construction of that function, which we show how to do in a restricted context.

To stay within the scope of a Master's thesis, we won't address other aspects of a logic adequate for natural language understanding like the representation of existence, non-monotonicity, the presuppositions, the implicatures, and the disambiguation of words.

Part I

Families of Sets and Quantification

Chapter 2

Theory of Generalized Quantifiers

In this chapter, we present the theory of generalized quantifiers for the study of quantification in natural language, as introduced by Barwise and Cooper (1981), who in turn attribute the initial notion to Montague (1974). Along the way, we also show that the transitive closure of subset relations between generalized quantifiers may be used for inferential purposes.

2.1 Introduction

2.1.1 Basic idea

Essentially generalized quantifiers separate the domain of quantification from the quantified predicate—as does natural language, but as opposed to standard quantification in first-order logic ($\forall$, $\exists$). The domain of quantification instead is part of the quantifier. By doing so, the meaning of a *simple* NP (a naked NP—without a possessive, a prepositional phrase, or a relative clause) corresponds to a single semantic unit—a generalized quantifier—hence preserving the syntactic structure at the semantic level, as motivated by the principle of compositionality (Montague, 1974). The preservation of the syntax is justified by an intuitive sense that language is closely related to the way we think. Therefore, the closer a representation follows language, the more likely it is to have similar properties. On that point, according to Partee, ter Meulen, and Wall (1993), the theory of generalized quantifiers has demonstrated its relevance.

> The theory of generalized quantifiers proves to be a new framework for the semantic explanation of linguistic data which have been studied extensively in syntactic theory. (p. 373)

As well, mimicking language might be crucial for the notion of obvious inferences (that which can be inferred easily). This is important since it is unthinkable to infer everything, and for language understanding, we need machines that infer easily what people infer easily. There are also more pragmatic justifications for such a treatment, namely to simplify the semantic interpretation and, as explained by Barwise and Cooper (1981), to handle determiners like *'most'*, which are not expressible in first-order logic. Finally, as has proven to be often true in mathematics, it is a good idea to see a problem from different perspectives in order to understand it.

2.1.2 Moore's representation

Before going into the theory, we present Moore's formalism (1981), since his notation is convenient for our later chapters. His formalism is derived from that of Barwise and Cooper; however, it stops at an intermediate step in the theory, since the quantifier and its domain are not regrouped in a single semantic unit. Instead, the quantifying structure has four parts: the determiner, the variable that is bound, the domain R of the variable, and the predicate P to be bound. For example:

(2.1) Every man is bald. $(\mathbf{every}; x; \mathtt{man}(x); \mathtt{bald}(x))$

(2.2) Some man is bald. $(\mathbf{some}; x; \mathtt{man}(x); \mathtt{bald}(x))$

(2.3) No man is bald. $(\mathbf{no}; x; \mathtt{man}(x); \mathtt{bald}(x))$

(2.4) At least two men are bald. $(\mathbf{atleast}_2; x; \mathtt{man}(x); \mathtt{bald}(x))$

Observe that the correspondence with English is more direct than it would be for a standard first-order logic notation. In particular, note from the formal definition of Moore's quantifying structure below, that the connective between R and P varies from $\rightarrow$, $\wedge$, and $\rightarrow\neg$, depending on the determiner.

$$(\mathbf{every}; x; R(x,\ldots); P(x,\ldots)) \triangleq \forall x,\ R(x,\ldots) \rightarrow P(x,\ldots); \tag{2.5}$$

$$(\mathbf{some}; x; R(x,\ldots); P(x,\ldots)) \triangleq \exists x,\ R(x,\ldots) \wedge P(x,\ldots); \tag{2.6}$$

$$(\mathbf{no}; x; R(x,\ldots); P(x,\ldots)) \triangleq \forall x,\ R(x,\ldots) \rightarrow \neg P(x,\ldots); \tag{2.7}$$

$$(\mathbf{atleast}_k; x; R(x,\ldots); P(x,\ldots)) \triangleq \exists_k x, R(x,\ldots) \wedge P(x,\ldots). \tag{2.8}$$

2.2 Formalization

2.2.1 Families of sets

Barwise and Cooper go further than Moore and integrate the domain of the quantification as part of the quantifiers, in order to achieve a semantic unit corresponding to a simple NP. Montague (1974) extracts the meaning of an NP from the meaning of the sentence using lambda abstraction. For example, the contribution of *'a man'* to the meaning of *'a man is bald'*, that is, $\exists x(\mathtt{man}(x) \wedge \mathtt{bald}(x))$, is extracted by abstracting the predicate $\mathtt{bald}$, yielding the lambda expression $\lambda P(\exists x(\mathtt{man}(x) \wedge$

$P(x)))$. Montague shows that the contribution to the meaning of an NP is given by such lambda expression.

(2.9) A man is bald. $\exists x(\mathtt{man}(x) \wedge \mathtt{bald}(x))$

(2.10) a man $\lambda P(\exists x(\mathtt{man}(x) \wedge P(x)))$

Moreover, since only one variable of the abstracted predicate P is bound by the lambda expression denoting a simple NP, its meaning can actually be expressed as a family of sets, more specifically, those sets which the lambda expression would map to **true** (see the set as the denotation of a unary predicate which is mapped by the lambda expression into a proposition whose denotation is true or false).[1] Hence, *'a man'* may be viewed as the family of sets which are mapped to the value **true** by the lambda expression $\lambda P(\exists x(\mathtt{man}(x) \wedge P(x)))$, that is, the family of every set that contains *'a man'*. Barwise and Cooper (1981) call such a family of sets a generalized quantifier. In general, the description for X (a given simple NP) of its family of sets is *'the family of every set that contains X'*. For instance, *'most men'* denotes the family of every set that contains *'most men'*; while, *'no men'* denotes the family of every set that contains *'no men'*.

2.2.2 Definitions

At this point, we define the main categories we will be discussing in this chapter and the following chapter. All the categories are defined with respect to $\mathcal{U}$, the domain.

Class A class is a subset of $\mathcal{U}$.

Quantifier A quantifier is a family of subsets of $\mathcal{U}$.

Determiner A determiner is a function that maps a class into a quantifier.

Relation A relation is a set of pairs of elements of $\mathcal{U}$.

Roughly, the class category is designed to denote common nouns[2]; the quantifier category denotes the simple NP's[3]; the determiner category is for determiners; and, the relation category is for transitive verbs. In natural language, the quantifier category and the determiner category happen to be further restricted, but we will only consider restrictions of interest for this thesis. In particular, determiners are restricted to functions which map a class into a quantifier that *lives on* that class, that is, a family of sets for which the membership of a set depends on the elements of the set belonging to

[1] An alternate argumentation is to observe that a determiner states that a given relationship holds between two sets (the domain of quantification and the quantified predicate), as in Moore's notation (1981), which relationship is expressible as a function mapping a set, the domain of quantification, into the family of sets for which the relationship holds (the quantifier).

[2] Actually, only those that denote a set like *'man'*, not those that denote a relation like *'mother'*.

[3] We will cover more complex NP's in the next chapters. For now, we consider only NP's with no possessive nor prepositional phrase and no anaphoric dependency.

that class. The idea is that the argument of the determiner D is the domain s of quantification, rendering elements outside that domain $(t \setminus s)$ are unimportant.

$$t \in D(s) \iff t \cap s \in D(s). \tag{2.11}$$

2.2.3 Some determiners

We proceed by defining some determiners. The determiner for *'every'* maps a set into the family of its supersets; the one for *'some'* maps a set into the family of sets not disjoint from it; the one for *'no'* maps a set into the family of sets disjoint from it; the one for *'at least k'* maps a set into the family of every set that contains *'at least k'* elements of the set; and so on...

$$\mathbf{every}(s) \triangleq \{\, x \subseteq \mathcal{U} \mid s \subseteq x \,\}; \tag{2.12}$$
$$\mathbf{some}(s) \triangleq \{\, x \subseteq \mathcal{U} \mid x \cap s \neq \emptyset \,\}; \tag{2.13}$$
$$\mathbf{no}(s) \triangleq \{\, x \subseteq \mathcal{U} \mid x \cap s = \emptyset \,\}; \tag{2.14}$$
$$\mathbf{atleast}_n(s) \triangleq \{\, x \subseteq \mathcal{U} \mid |s \cap x| \geq n \,\}; \tag{2.15}$$
$$\mathbf{atmost}_n(s) \triangleq \{\, x \subseteq \mathcal{U} \mid |s \cap x| \leq n \,\}; \tag{2.16}$$
$$\mathbf{exactly}_n(s) \triangleq \{\, x \subseteq \mathcal{U} \mid |s \cap x| = n \,\}; \tag{2.17}$$
$$\mathbf{most}(s) \triangleq \{\, x \subseteq \mathcal{U} \mid |s \cap x| \geq \frac{|s|}{2} \,\}. \tag{2.18}$$

2.2.4 Very simple sentences

At this point, it is possible to handle some very simple sentences. Given that the denotation of `man` is the set of *'all men'*, and `bald`, the set of *'all things that are bald'*,[4] then the denotation of *'every man is bald'* is `bald` $\in$ **every**(`man`). It follows since `bald` $\in$ **every**(`man`) if and only if `bald` is a superset of `man`, that is, if *'every man is bald'*. Similarly, the other interpretations below are valid. (This is so since the family of sets was obtained from the lambda expression in such a way that testing for membership corresponds to applying the lambda expression.) Note that proper nouns are treated as quantifiers too: for instance, *'John'* denotes the family of every set that includes the singleton `john`, that is, **every**(`john`) or **some**(`john`), which in that case are equal.

(2.19) Every man is bald. — `bald` $\in$ **every**(`man`)

(2.20) Some man is bald. — `bald` $\in$ **some**(`man`)

(2.21) Q is P. — $P \in Q$

[4] As a convention, for the denotation of a common noun, a proper noun, an adjective, or a verb, we use its root form in this special font. The denotation is either a subset of $\mathcal{U}$ or of $\mathcal{U} \times \mathcal{U}$ depending on the category of the word.

(2.22) John is bald. $\texttt{bald} \in \textbf{every}(\texttt{john})$

2.2.5 Correspondence with Moore's notation

The membership test above can be rewritten with Moore's notation. (Note that in Moore's notation, the determiner determines if a given relationship holds between two sets, whereas here, a determiner maps a set into a family of sets. Yet, these two notions are equivalent, since there is a bijection between the two, $R_D(A, B) \Longleftrightarrow A \in f_D(B)$.)

(2.23) $$(D; x; R(x, \ldots); P(x, \ldots)) \iff \{\, x \mid P(x, \ldots) \,\} \in D(\{\, x \mid R(x, \ldots) \,\}).$$

2.3 Quantifiers

To study quantifiers, Barwise and Cooper (1981) borrowed standard mathematics tools such as set complement, intersection, union, and the notion of monotonicity. Monotonicity turns out to be an important inferential notion; for instance, see Purdy (1991). At the end of this section, we propose to capture monotonic inferences through the transitive closure of the inclusion, as we make a simple observation that a simple sentence can be expressed as a subset relationship between two monotone quantifiers.

2.3.1 Negation and duality

There are several meaningful ways to produce a new quantifier from a given one.[5]

(2.24) $$\neg Q = \{\, x \subseteq \mathcal{U} \mid x \notin Q \,\};$$

(2.25) $$Q\neg = \{\, x \subseteq \mathcal{U} \mid \overline{x} \in Q \,\};$$

(2.26) $$Q^{\sim} = \{\, x \subseteq \mathcal{U} \mid \overline{x} \notin Q \,\}.$$

Given a quantifier Q, its complement, $\neg Q$, corresponds to sentence negation in example (2.27); whereas, the set of complements, $Q\neg$, corresponds to verb phrase negation in example (2.28). The combination, that is, the complement of the sets of complements or dual, $Q^{\sim}$, is interesting for mathematical purposes.

(2.27) It is not true that some men are bald. $\texttt{bald} \in \neg\textbf{some}(\texttt{man})$

(2.28) Some men are not bald. $\texttt{bald} \in \textbf{some}(\texttt{man})\neg$

[5] $\overline{x}$ is defined as the complement of x.

With regards to the quantifiers that we have already defined, the complement operators relates them in the following way.

$$\neg\mathbf{some}(s) = \mathbf{no}(s); \tag{2.29}$$

$$\mathbf{every}(s)\neg = \mathbf{no}(s); \tag{2.30}$$

$$\mathbf{every}(s)^{\sim} = \mathbf{some}(s); \tag{2.31}$$

$$\neg\mathbf{atleast}_k(s) = \mathbf{atmost}_{k-1}(s). \tag{2.32}$$

The effect of the complement operators on membership is derived from their definition.

$$t \in Q \Longleftrightarrow \bar{t} \in Q\neg \Longleftrightarrow t \notin \neg Q \Longleftrightarrow \bar{t} \notin Q^{\sim}. \tag{2.33}$$

Note also the effect of the dual operator on subset relations.

$$Q_1 \subseteq Q_2 \Longleftrightarrow Q_1{}^{\sim} \supseteq Q_2{}^{\sim}. \tag{2.34}$$

Other properties of these operators are given in appendix A.

2.3.2 Intersection and Union

Another means of producing new quantifiers from previous ones is using set intersection and union. These correspond to conjunction and disjunction.

$$P \in Q_1 \wedge P \in Q_2 \iff P \in Q_1 \cap Q_2; \tag{2.35}$$

$$P \in Q_2 \vee P \in Q_2 \iff P \in Q_1 \cup Q_2. \tag{2.36}$$

2.3.3 Monotone quantifiers

The mathematical notion of monotonicity is central for inference. A monotone increasing (decreasing) quantifier has the property that if it contains a set, it also contains all its supersets (subsets). We use $\mathcal{Q}_+$ and $\mathcal{Q}_-$ to denote the family of monotone increasing and decreasing quantifiers.

$$\mathcal{Q}_+ = \{ Q \subseteq \mathcal{P}ow(\mathcal{U}) \mid \forall s \subseteq t \subseteq \mathcal{U},\ s \in Q \to t \in Q \}; \tag{2.37}$$

$$\mathcal{Q}_- = \{ Q \subseteq \mathcal{P}ow(\mathcal{U}) \mid \forall s \subseteq t \subseteq \mathcal{U},\ t \in Q \to s \in Q \}. \tag{2.38}$$

The interest of this notion for inference follows from the possibility of inferring membership in a quantifier from the membership of a superset (or subset). Moreover, Barwise and Cooper (1981, p. 187) observed and proposed as a universal for natural language the so-called monotonicity constraint, stated below, which stresses the importance of monotone quantifiers in natural language.

Monotonicity constraint. The simple NP's of any natural language express monotone quantifiers or conjunctions of monotone quantifiers.

(Note that simple NP's do not include NP's formed out of complex determiners like *'an even number of'* which are not monotone.)

Among the quantifiers we have defined, **every**(s), **some**(s), $\mathbf{atleast}_k(s)$, and **most**(s) are monotone increasing quantifiers, while **no**(s) and $\mathbf{atmost}_k(s)$ are monotone decreasing for any s; see (Barwise and Cooper, 1981).

Barwise and Cooper also give the following properties of monotone quantifiers: the complement operators (except the dual) map a monotone quantifier into one of inverse polarity (from increasing to decreasing, or vice-versa), while the dual operator preserves the polarity (since it is switched twice). Intersection and union also preserve the polarity when combining two monotone quantifiers with the same polarity.

2.3.4 Principle of monotone quantifiers

Since a monotone increasing quantifier contains all the supersets of a set it contains (say t), it includes the family of all the supersets of t, that is, by definition, the quantifier **every**(t); conversely, if a monotone includes **every**(t), it contains t since t is itself one of its supersets. The analog for monotone decreasing quantifiers is obtained by reduction to the monotone increasing case through complementation ($t \in Q_- \iff t \notin \neg Q_-$).

Given $Q_+ \in \mathcal{Q}_+$ and $Q_- \in \mathcal{Q}_-$:

$$t \in Q_+ \iff \mathbf{every}(t) \subseteq Q_+; \tag{2.39}$$

$$t \in Q_- \iff \mathbf{every}(t) \not\subseteq \neg Q_-. \tag{2.40}$$

This observation is of interest because representing membership using subset relations allows the transitivity of inclusion to do some types of inferences, that is, to derive new subset relationships (or membership relationship).[6]

(2.41) John is a man. $\mathbf{every}(\texttt{john}) \supseteq \mathbf{every}(\texttt{man})$

(2.42) Every man is bald. $\mathbf{every}(\texttt{man}) \supseteq \mathbf{every}(\texttt{bald})$

(2.43) John is bald. $\mathbf{every}(\texttt{john}) \supseteq \mathbf{every}(\texttt{bald})$

In this example, (2.43) follows from (2.41) and (2.42) by transitivity.[7] Note that this kind of inference is not restricted to propernouns like `john`, and works as well with other quantifiers, like **some**(`man`).

[6] In this example and from now on, $\supseteq$ is used instead of $\subseteq$ (with the opposite meaning) to preserve the order of English in our notation.

[7] For the moment, *'John is a man'* is treated as *'John is man'* where *'man'* is a predicate like *'bald'*.

This suggests a formalism based on subset and non-subset relations between monotone increasing quantifiers captures; it is in part the subject of the next chapter. (The prominence of monotone quantifiers in natural language justifies a formalism bias towards them.) For this formalism, it becomes interesting to understand the meaning of subset relations between monotone increasing quantifiers like $\mathbf{atleast}_i(s) \subseteq \mathbf{atleast}_j(t)$.

2.4 Determiners

A determiner maps a set into a quantifier. In this section, we look at the properties of **every** and **some** needed for the next chapter, at the classes of determiners that are of interest for this thesis, and present a lemma useful in figuring out the meaning of certain subset relations between monotone increasing quantifiers.

2.4.1 Every and Some

Since the next chapter is concerned with a logic based on subset relations of quantifiers formed with **every** and **some**, we now look at the signification of subset relations between these types of quantifiers. (These are generally known in the literature, and are proven in appendix A.)

The first type of subset relationship occurs when the determiners are identical. Note that the subset relationship at the class level is preserved (or reversed) at the quantifier level, Barwise and Cooper (1981) called this property persistence (anti-persistence).[8]

$$\mathbf{some}(s) \subseteq \mathbf{some}(t) \iff s \subseteq t; \tag{2.44}$$

$$\mathbf{every}(s) \subseteq \mathbf{every}(t) \iff t \subseteq s. \tag{2.45}$$

The second type of subset relationship is when the restriction sets are the same. In this case, quantifiers reveal their counting ability.

$$\mathbf{every}(s) \subseteq \mathbf{some}(s) \iff |s| \geq 1; \tag{2.46}$$

$$\mathbf{some}(s) \subseteq \mathbf{every}(s) \iff |s| \leq 1. \tag{2.47}$$

For the last type, both the determiners and the sets are different.

$$\mathbf{every}(s) \subseteq \mathbf{some}(t) \iff |s \cap t| \geq 1; \tag{2.48}$$

[8] In category theory, this would induce a functor between the quantifier category and the class category. Note that it suggests a formalism where some types of inferences are done by transitivity at the class level and others at the quantifier level, and where the information inferred at each level is communicated at the other level through these functors.

$$(2.49)\qquad \mathbf{some}(s) \subseteq \mathbf{every}(t) \iff |s \cup t| \leq 1 \lor s = \emptyset \lor t = \emptyset.$$

Finally, these observations concerns the empty set and are quite obvious.

$$(2.50)\qquad \mathbf{some}(\emptyset) = \emptyset;$$

$$(2.51)\qquad \mathbf{every}(\emptyset) = \mathcal{P}ow(\mathcal{U}).$$

2.4.2 Intersection lemma

The intersection lemma is of interest in order to understand subset relationships between quantifiers. It is included as evidence for the viability of a system based on the subset relations between quantifiers. Before introducing the intersection lemma, we present some standard classes of determiners (elements of $\mathcal{D}$).

$$(2.52)\qquad \mathcal{D} = \{\, D\colon \mathcal{P}ow(\mathcal{U}) \to \mathcal{P}ow(\mathcal{P}ow(\mathcal{U})) \mid \forall x, y \subseteq \mathcal{U},\ x \cap y \in D(y) \to x \in D(y) \,\};$$

$$(2.53)\qquad \mathcal{D}_+ = \{\, D \in \mathcal{D} \mid \forall x \subseteq \mathcal{U},\ D(x) \in \mathcal{Q}_+ \,\};$$

$$(2.54)\qquad \mathcal{D}_- = \{\, D \in \mathcal{D} \mid \forall x \subseteq \mathcal{U},\ D(x) \in \mathcal{Q}_- \,\};$$

$$(2.55)\qquad \mathcal{D}_{\text{SYM}} = \{\, D \in \mathcal{D} \mid \forall x, y \subseteq \mathcal{U},\ x \in D(y) \leftrightarrow y \in D(x) \,\}.$$

(Note that $\mathcal{D}$ is restricted to include only determiners that output quantifiers that live on the restriction set.[9]) The class $\mathcal{D}_+$ ($\mathcal{D}_-$) contains every determiner that outputs only monotone increasing (decreasing) quantifiers. The last class of determiners, $\mathcal{D}_{\text{SYM}}$, has also been studied extensively (Barwise and Cooper, 1981). It corresponds to determiners that are symmetric, those for which the restriction set can be interchanged with the set quantified ($t \in D(s) \iff s \in D(t)$). The intersection lemma shown below is actually a generalization of a property of monotone increasing symmetric determiners (say $D \in \mathcal{D}_+ \cap \mathcal{D}_{\text{SYM}}$) observed by Barwise and Cooper (1981), that is, that $t \in D(s) \iff s \cap t \in D(s \cap t)$.

Before stating the intersection lemma, we introduce two new classes, $\mathcal{D}_S$ and $\mathcal{D}_E$, designed in a such way that the lemma is true. However, their description in English is not clear; they have to be understood through their properties. The key property of determiners in $\mathcal{D}_S$ or $\mathcal{D}_E$ (say D) is the possibility of constructing $D(s)$ from the set s and the quantifier $D(t)$ for any t that includes s. The construction method depends on the class ($\mathcal{D}_S$ or $\mathcal{D}_E$) that D is from; the method used in $\mathcal{D}_S$ works for **some**, hence its name, and the one for $\mathcal{D}_E$ works for **every**.

$$\mathcal{D}_S = \{\, D \in \mathcal{D} \mid \forall s \subseteq t \subseteq \mathcal{U},\ \forall x \in \mathcal{U},\ [x \in D(s) \leftrightarrow \forall y \in \mathcal{U},$$

[9] Natural language determiners are subject to other constraints, but they are not of interest here; see (Barwise and Cooper, 1981; Benthem, 1984; Westerståhl, 1984).

$$x \cap s = y \cap s \to y \in D(t)] \};\tag{2.56}$$

$$\mathcal{D}_E = \{ D \in \mathcal{D} \mid \forall s \subseteq t \subseteq \mathcal{U},\ \forall x \in \mathcal{U},\ [x \in D(s) \leftrightarrow \exists y \in \mathcal{U},$$
$$x \cap s = y \cap s \wedge y \in D(t)] \}.\tag{2.57}$$

In appendix A, we show some properties of these two classes of determiners. Notably, **some** and $\mathbf{atleast}_k$ belong to $\mathcal{D}_S$, while **every** belongs to $\mathcal{D}_E$. Also, monotone increasing determiners are in $\mathcal{D}_S$ if and only if they are symmetric (in $\mathcal{D}_{\text{SYM}}$). Yet, both classes of determiners are distinct. Moreover, $\mathcal{D}_S$ and $\mathcal{D}_E$ are in some sense dual, since the dual[10] of every determiner of $\mathcal{D}_S$ is in $\mathcal{D}_E$, and vice-versa. Finally, determiners in $\mathcal{D}_S$ ($\mathcal{D}_E$) preserve (invert) subset relationships of the class level at the quantifier level.

$$S \in \mathcal{D}_S, E \in \mathcal{D}_E \text{ and } s \subseteq t \implies S(s) \subseteq S(t) \text{ and } E(t) \subseteq E(s).\tag{2.58}$$

The intersection lemma is stated as follows, given $S \in \mathcal{D}_S$ and $E \in \mathcal{D}_E$:

$$S(s) \supseteq E(t) \iff S(s \cap t) \supseteq E(s \cap t).\tag{2.59}$$

Its corollaries are:

$$S(s) \supseteq E(t) \iff S(t) \supseteq E(s);\tag{2.60}$$
$$S(s) \supseteq E(t) \implies S(t) \supseteq E(t);\tag{2.61}$$
$$S(s) \supseteq E(t) \implies S(s) \supseteq E(s).\tag{2.62}$$

This lemma is interesting because the meaning of $S(s \cap t) \supseteq E(s \cap t)$ happens to be easier to determine then the one of $S(s) \supseteq E(t)$ since both determiners map the same set. (The lemma is used in appendix A to prove subset relationships involving $\mathbf{atleast}_k$ that would not have been trivial to prove otherwise.)

2.5 Relations

This section is not directly related to generalized quantifiers, but acts as a transition for the next chapter.

So far, we considered only simple cases where the restriction set of a quantifier is a set, or unary predicate, and the predicate being quantified is also a unary predicate. In natural language, however, this is not sufficient; both predicates may have higher arity. Barwise and Cooper (1981)

[10] For any given input, the dual of a determiner outputs the dual of the quantifier outputted by the original determiner.

allow arbitrary arity for both predicates through the use of variables in the standard fashion—both predicates are still seen as sets but whose value is determined by the value assigned to some variables.

Here, instead, we follow McAllester and Givan (1992) and allow a binary predicate to be bound by a quantifier, but still require a unary predicate for the restriction set. (This is not sufficient for natural language but, as discussed in the next chapter, has interesting properties.) To do so, we define two new operators: $\mathcal{RQ}$, to quantify a binary predicate; Π, to simplify the definition of the first one.

Given $d \in \mathcal{U}$, $R \subseteq \mathcal{U} \times \mathcal{U}$ and $Q \subseteq \mathcal{P}ow(\mathcal{U})$):

$$\Pi_d(R) \quad = \quad \{\, d' \subseteq \mathcal{U} \mid (d, d') \in R \,\}; \tag{2.63}$$

$$\mathcal{RQ}_R(Q) \quad = \quad \{\, d \subseteq \mathcal{U} \mid \Pi_d(R) \in Q \,\}. \tag{2.64}$$

Formally, given that R is a binary relation, and Q, a quantifier; R is treated as a set which depends on a variable, the set of values for its second argument it relates to a given value of its first argument (the operator Π is used for that metamorphosis); and, $\mathcal{RQ}_R(Q)$ is the set of bindings for which the R is contained in Q.

This modification permits handling simple SVO (subject–verb–object) sentences like *'every man loves a woman'*. In some sense, the sentence is understood as *'every man is such that a woman is loved by him'*. The Π operator serves to construct the set of entities *'loved by him'*, that is, $\Pi_d(\texttt{love})$ $(= \{\, d' \in \mathcal{U} \mid (d, d') \in \texttt{love} \,\})$ where the variable d corresponds to *'him'*; then the $\mathcal{RQ}$ operator constructs the set of entities *'such that a woman is loved by him'* by computing the set of bindings for *'him'*, that is, $\mathcal{RQ}_{\texttt{love}}(\textbf{some}(\texttt{woman}))$ $(= \{\, d \in \mathcal{U} \mid \Pi_d(\texttt{love}) \in \textbf{some}(\texttt{woman}) \,\})$; from this last set, the meaning of the full sentence is derived as usual.

(2.65) Every man loves a woman. $\quad \textbf{every}(\texttt{man}) \supseteq \textbf{every}(\mathcal{RQ}_{\texttt{love}}(\textbf{some}(\texttt{woman})))$

The following properties of $\mathcal{RQ}$ are used in the next chapter.

$$\mathcal{RQ}_R(\emptyset) \quad = \quad \emptyset; \tag{2.66}$$

$$\mathcal{RQ}_R(\mathcal{P}ow(\mathcal{U})) \quad = \quad \mathcal{U}; \tag{2.67}$$

$$Q_1 \subseteq Q_2 \quad \Longrightarrow \quad \mathcal{RQ}_R(Q_1) \subseteq \mathcal{RQ}_R(Q_2). \tag{2.68}$$

Chapter 3

Montagovian Syntax

In this chapter, we look at the so-called Montagovian syntax of McAllester and Givan (1992), whose *variable-free*[1] version quantifies only unary or binary predicates and restricts the domain of quantification to unary predicates. We show that the variable-free Montagovian syntax is isomorphic to a logic based on subset relations between quantifiers (referred as the BGQ logic), thereby demonstrating the feasibility and the potential of such logic.

The virtue of the variable-free Montagovian syntax is that satisfiability is decidable in polynomial time (with respect to the number of formulas), if specific conditions are satisfied: that every class involved is known to be empty or not. This leads to a principled notion of obviousness, rather than a definition through arbitrary time constraints on the inference mechanism.

The interest of the variable-free Montagovian syntax for natural language is not directly obvious, since it only deals with a small fragment of natural language. In fact, *'every'* and *'some'* (and *'no'*, obtained by negation) are the only determiners, and only simple SVO (subject–verb–object) sentences and simple NP's with an optional *'of'* prepositional phrase (with narrow scope) are expressible, no other propositional phrases, no adjectives, no relative clauses, no anaphora, and so on. Yet, its syntax mimics natural language, and the notion of obviousness it offers might serve as a basis for investigating what people infer easily. Most especially, its similarity with natural language may be reflected in its inferential properties. Moreover, the condition by which satisfiability is in polynomial time has, we propose, a natural language counterpart, since in general, the classes mentioned are presupposed to be non-empty unless stated otherwise.

For example, in the sentence below, it is presupposed that *'there are men on this island'* until explicitly contradicted.

(3.1) No man on this island is bald; (because there is no man on this island).

[1]This basically means 'with no lambda-like expressions'; McAllester and Givan use instead the expression *quantifier-free*, as they see this as the essence of quantification, but we prefer the expression *variable-free* to avoid any confusion with our notion of quantifier.

Also, in natural language, sets denoted by adjectives like *'bald'*, adjectival clauses like *'who loves Mary'*, or verb phrases like *'... loves Mary'* can always be assumed to be non-empty without affecting consistency, since they cannot serve by themselves as domain of quantification (as opposed to common nouns or their intersection with common nouns). Consequently, for each of these sets, it can be assumed that a non-quantifiable object[2] belongs to them and no contradiction would be derivable. For instance, the following sentences do not imply that *'bald'* denotes the empty set, since something that is not a 'thing' may be bald!

(3.2) Nothing is bald.

(3.3) It is not true that a bald thing exists.

(3.4) No man is bald; everything that is not a man is not bald.

On the practical side, to see the variable-free Montagovian syntax through subset relations between quantifiers may permit a slightly more efficient implementation. Also, it has the advantage of offering a smaller set of simpler inference rules—most rules are of the type 'from A infer B' where A is a simple formula instead of a conjunction of formulas. Finally, it offers new mathematical tools to pursue the analysis of the Montagovian syntax and investigate possible extensions.

3.1 The variable-free fragment

3.1.1 General idea

McAllester and Givan (1992) introduced the Montagovian syntax in the context of interactive theorem provers (which prove the *obvious* steps automatically, but ask for human assistance for the more subtle steps). To capture the notion of obviousness, they propose the idea of local inferences (McAllester, Givan and Fatima, 1989; Givan, McAllester and Shalaby, 1991; McAllester and Givan, 1992; McAllester, 1993; McAllester and Givan, 1993). Their idea is to restrict the formulas generated by the inference mechanism to formulas made up of subexpressions of the original formulas. In the case of the variable-free Montagovian syntax this means formulas made up of class expressions already present. Consequently, local inferences bound the inference mechanism to polynomial time since the number of derivable formulas is polynomial.[3]

The Montagovian syntax was designed with the belief that natural language has an effective syntax for inference that a syntax similar to it would inherit. It is named after Montague (1974), as it tries to preserve the syntactic structure of natural language. In particular, the domain of quantification is specified explicitly.

[2] An object that does not belong to any possible domain of quantification, that is, not in the denotation of any common noun.

[3] It is in fact quadratic with the number of classes, since a formula is made of at most two classes and there is just a finite number of ways they can be combined.

For its expressive power, the variable-free Montagovian syntax lies between propositional logic and first-order logic. However, the full Montagovian syntax, that is, the variable-free fragment plus lambda abstraction and the formula connectives *'and'* and *'or'*, is equivalent to first-order logic.

3.1.2 Syntax and semantics

The variable-free Montagovian syntax is defined recursively through two types of objects, class expressions and formulas.

Class expressions

- a monadic predicate symbol;
- (R (some s)) or (R (every s)) where R is a binary predicate symbol and s is a class expression.

Formulas

- (at-most-one s), $\exists s$, (some s w) or (every s w) where s and w are class expressions;
- $\neg F$ where F is a formula.

The class expressions denote classes of objects. We can make a crude analogy with natural language—though even for the fragment of English it covers, the Montagovian syntax does not capture the subtleties of natural language. The analog of a monadic predicate symbol is a common noun (one denoting a set like the sets of *'men'*, not a relation like *'mother of'*); whereas, the expressions (R (some s)) and (R (every s)) are the analog of verb phrases, where R is the verb and (some s) or (every s), the complement. They can also been seen as the analog of a common noun R (one denoting a relation) modified by a narrow-scope *'of'* prepositional phrase with (some s) or (every s) as the complement. However, these are not common in natural language—except when it does not matter because the complement denotes a constant—since *'of'* prepositional phrases tend to take wide-scope over the NP they modify. The formulas (at-most-one s) and $\exists s$ state that a class expression denotes a set of at most one element and at least one element, respectively. In the formulas (some s w) or (every s w) s denotes the domain of quantification and w the quantified predicate. They are the analog of sentences, where the subject is (some s ...) or (every s ...) and the verb phrase is w (w denoting the set of elements satisfying the verb phrase). Finally, a formula can be negated, which corresponds to sentence negation (*'it is not true that ...'*).

In this example, a simple English sentence is expressed in the variable-free Montagovian syntax, with its logically equivalent representation in a standard first-order logic notation.

(3.5) Every man loves a woman. (every $\texttt{man}$ ($\texttt{love}$ (some $\texttt{woman}$)))

$$\forall x,\ \texttt{man}(x) \rightarrow (\exists y,\ \texttt{woman}(y) \wedge \texttt{love}(x, y))$$

Formally, the meaning of the expressions above is given by the function φ, defined recursively, which maps every expression into its denotation. The terminal symbol for a class expression is a monadic predicate, which φ maps into a set of elements. As well, the symbol referred by R in the expressions $(R\ (\text{some}\ s))$ and $(R\ (\text{every}\ s))$ is a binary predicate, which is mapped by φ into a set of pairs of elements.

$$\varphi((R\ (\text{every}\ s))) = \{\, d \in \mathcal{U} \mid \forall d' \in \varphi(s),\ (d, d') \in \varphi(R)\,\}; \tag{3.6}$$

$$\varphi((R\ (\text{some}\ s))) = \{\, d \in \mathcal{U} \mid \exists d' \in \varphi(s),\ (d, d') \in \varphi(R)\,\}; \tag{3.7}$$

$$\varphi((\text{every}\ s\ t)) = \textbf{true} \text{ if } \varphi(s) \subseteq \varphi(t), \textbf{false}\text{otherwise}; \tag{3.8}$$

$$\varphi((\text{some}\ s\ t)) = \textbf{true} \text{ if } \varphi(s) \cap \varphi(t) \neq \emptyset, \textbf{false}\text{otherwise}; \tag{3.9}$$

$$\varphi(\exists s) = \textbf{true} \text{ if } \varphi(s) \neq \emptyset, \textbf{false}\text{otherwise}; \tag{3.10}$$

$$\varphi((\text{at-most-one}\ s)) = \textbf{true} \text{ if } |\varphi(s)| \leq 1, \textbf{false}\text{otherwise}; \tag{3.11}$$

$$\varphi(\neg\Psi) = \textbf{true} \text{ if } \varphi(\Psi) \neq \textbf{true}, \textbf{false}\text{otherwise}. \tag{3.12}$$

McAllester and Givan also provide a set of inference rules for the variable-free Montagovian syntax, which is reproduced in appendix B.

3.1.3 Satisfiability

McAllester and Givan (1992) show that deciding the satisfiability of a set of variable-free Montagovian formulas is NP-complete. However, they demonstrate it can be done in polynomial time given that the class expressions of each formula are known to denote the empty set or not. (As suggested at the beginning of the chapter, this requirement is psycholinguistically plausible, and it is our belief that it should be investigated further.)

More precisely, local inferences are complete for the satisfiability problem given that for every class expression s of every formulae, $\exists s$ or $\neg\exists s$ is locally derivable. This implies polynomial time since the number of class expressions in the original set of formulas is bounded by their total size, so the number of formulas made of these class expressions is quadratic. Therefore, the derivation process is guaranteed to stop after a quadratic number of derivations (since after that there is nothing more to prove); and since checking every inference rule to find a new derivation is done in polynomial time, the whole process is bounded by polynomial time.

3.2 BGQ logic

At this point, we propose to transpose the Montagovian syntax into a logic based on generalized quantifiers, that we call the basic generalized quantifier logic (BGQ logic). It is done easily, since the variable-free Montagovian syntax specifies explicitly the domain of quantification—one needs only to regroup the determiner and the domain of quantification in a single semantic unit. In this logic, the formulas of the variable-free Montagovian syntax are expressed as subset relations between generalized quantifiers. As a result, we show that this new representation reduces both the number and complexity of inference rules.

3.2.1 Syntax and semantics

The BGQ logic uses the same two objects as the Montagovian syntax, as well as a third for the quantifiers that are extracted from the class expressions and the formulas.

Class expressions

- a monadic predicate symbol;
- $\langle R\ Q \rangle$, where R is a binary relation symbol and Q is a quantifier.

Quantifier expressions

- [some s] or [every s], where s is a class expression.

Formulas

- $Q_1 \sqsupseteq Q_2$ or $Q_1 \not\sqsupseteq Q_2$, where Q_1 and Q_2 are quantifier expressions.

Each BGQ expression has an analog in the variable-free Montagovian syntax, which can be computed recursively using the following table.

$$(R \text{ (some } s)) \overset{\text{corresponds}}{\longleftrightarrow} \langle R \text{ [some } s] \rangle; \tag{3.13}$$

$$(R \text{ (every } s)) \overset{\text{corresponds}}{\longleftrightarrow} \langle R \text{ [every } s] \rangle; \tag{3.14}$$

$$(\text{every } s\ t) \overset{\text{corresponds}}{\longleftrightarrow} [\text{every } s] \sqsupseteq [\text{every } t]; \tag{3.15}$$

$$(\text{some } s\ t) \overset{\text{corresponds}}{\longleftrightarrow} [\text{some } s] \sqsupseteq [\text{every } t]; \tag{3.16}$$

$$\exists s \overset{\text{corresponds}}{\longleftrightarrow} [\text{some } s] \sqsupseteq [\text{every } s]; \tag{3.17}$$

$$(\text{at-most-one } s) \overset{\text{corresponds}}{\longleftrightarrow} [\text{every } s] \sqsupseteq [\text{some } s]. \tag{3.18}$$

For example, the representation of the following sentence can be translated back and forth between the BGQ logic and the variable-free Montagovian syntax.

(3.19) Every man loves a woman. $\qquad [\text{every } \mathtt{man}] \sqsupseteq [\text{every } \langle \mathtt{love}\ [\text{some } \mathtt{woman}] \rangle]$

$(\text{every } \mathtt{man}\ (\mathtt{love}\ (\text{some } \mathtt{woman})))$

The semantics of the BGQ logic are defined using the operator and the determiner functions defined in the previous chapter. Yet, we show in appendix B that the corresponding expressions in the two representations receive the same semantics.

$$\varphi([\text{some } s]) = \mathbf{some}(\varphi(s)); \tag{3.20}$$

$$\varphi([\text{every } s]) = \mathbf{every}(\varphi(s)); \tag{3.21}$$

$$\varphi(\langle R\ Q\rangle) = \mathcal{RQ}_{\varphi(R)}(\varphi(Q)); \tag{3.22}$$

$$\varphi(Q_1 \sqsupseteq Q_2) = \mathbf{true} \text{ if } \varphi(Q_2) \subseteq \varphi(Q_1), \mathbf{false}\text{otherwise}; \tag{3.23}$$

$$\varphi(Q_1 \not\sqsupseteq Q_2) = \mathbf{true} \text{ if } \varphi(Q_2) \not\subseteq \varphi(Q_1), \mathbf{false}\text{otherwise}. \tag{3.24}$$

3.2.2 Inference rules

The difference between the BGQ logic and the variable-free Montagovian may be seen as purely syntactic. Yet, it permits the factorization of the 16 inference rules of the variable-free Montagovian into 10 rules, which are significantly simpler than the original ones. (In the variable-free Montagovian syntax, 8 inference rules involve the conjunction of two or more formulas as precondition; whereas the BGQ logic needs only two such rules; one of them being trivial (the one that raises a contradiction); the only remaining complex rule is the transitivity rule.)

The inference rules are shown to be sound in appendix B. The rule (3.25) expresses that a constant c is a singleton. The rules (3.26), (3.27), and (3.28) axiomatize the subset relation. And the rule (3.29) encodes the effect of the dual operator on subset relations. Furthermore, the rule (3.30) encodes the property of the $\mathcal{RQ}$ operator to preserve subset relations between quantifiers. The remaining rules are direct translations of their counterparts in the Montagovian syntax. They capture respectively that there is no bigger quantifier than the universal quantifier ($\mathcal{P}ow(\mathcal{U})$), the $\mathcal{RQ}$ operator maps the empty quantifier into the empty set, and the universal quantifier into the universal set ($\mathcal{U}$).

$$\vdash [\text{some } c] \sqsupseteq [\text{every } c] \sqsupseteq [\text{some } c]; \tag{3.25}$$

$$\vdash Q \sqsupseteq Q; \tag{3.26}$$

$$Q_1 \sqsupseteq Q_2,\ Q_2 \sqsupseteq Q_3 \vdash Q_1 \sqsupseteq Q_3; \tag{3.27}$$

$$Q_1 \sqsupseteq Q_2,\ Q_1 \not\sqsupseteq Q_2 \vdash \mathbf{false}; \tag{3.28}$$

$$Q_1 \sqsupseteq Q_2 \vdash Q_2{}^{\sim} \sqsupseteq Q_1{}^{\sim}; \tag{3.29}$$

$$Q_1 \sqsupseteq Q_2 \vdash [\text{some } \langle R\ Q_1\rangle] \sqsupseteq [\text{some } \langle R\ Q_2\rangle]; \tag{3.30}$$

$$[\text{every } r] \not\sqsupseteq [\text{every } t] \vdash [\text{some } r] \sqsupseteq [\text{every } r]; \tag{3.31}$$

$$[\text{some } \langle R\ [\text{some } s]\rangle] \sqsupseteq [\text{every } \langle R\ [\text{some } s]\rangle] \vdash [\text{some } s] \sqsupseteq [\text{every } s]; \tag{3.32}$$

(3.33) $$[\text{some } s] \not\sqsupseteq [\text{every } s] \quad \vdash \quad [\text{every } t] \sqsupseteq [\text{every } \langle R\, [\text{every } s]\rangle].$$

(Note that $Q^{\sim}$ is seen for now as a notational convenience: $[\text{every } s]^{\sim}$ means [some s], and vice-versa; so the rule for duality is a schema of rules.)

3.2.3 Equivalence to the variable-free Montagovian syntax

The BGQ logic and the variable-free Montagovian syntax are equivalent. We give here a schema of the proof; see appendix B for the missing parts. The proof is direct, although tedious. First, we have a bijection between formulas of the two formalisms. Technically, the only complication is that the BGQ formula [every s] $\sqsupseteq$ [some t], whose meaning is $s = \emptyset \vee t = \emptyset \vee |s \cup t| \leq 1$, has no equivalent in the variable-free Montagovian syntax because of the disjunctions. However, since the class expressions are required to be known to denote the empty set or not in order to apply the theorem on satisfiability, the problematic formula is trivially true if $s = \emptyset$ or $t = \emptyset$ or means that $s = t$ and $|s| = 1$, which is expressible in the variable-free Montagovian syntax as (every $s\ t$)$\wedge$(every $t\ s$)$\wedge$(at-most-one s)$\wedge \exists s$. The second step is to show that the bijection preserves the semantics. The third step is to demonstrate that the inference rules of the BGQ logic are sound. The last step is to show that the inference rules of the BGQ logic can *locally* infer every formula that the inference rules of the variable-free Montagovian syntax can locally infer. This is done by proving that for every single rule of the variable-free Montagovian syntax, there is a sequence of rules in the BGQ logic which can produce the same result (without introducing a new class expression). (Note that local inferences in the BGQ logic are restricted to formulas formed of class expressions already present, that is, given that s and t are two such class expressions, $[D\ s] \sqsupseteq [D'\ t]$ or $[D\ s] \not\sqsupseteq [D'\ t]$, where D and $D' \in \{\text{every}, \text{some}\}$.)

Therefore, the inference rules of the BGQ logic are complete for the satisfiability problem, given that for every class expression s, [some s] $\sqsupseteq$ [every s] or [some s] $\not\sqsupseteq$ [every s] is locally derivable.

Part II

Binary Relations and Anaphora

Chapter 4

An Extended Relation Algebra

4.1 Introduction

In the two previous chapters, we investigated the issue of quantification in a variable-free context. The variable-free Montagovian syntax and the BGQ logic cannot represent simple sentences involving anaphora such as sentence (4.1).

(4.1) Every man_1 loves his_1 wife.

The traditional way to account for such anaphora is by using variables or lambda abstraction. Barwise and Cooper (1981) adopted this method, as did McAllester and Givan (1992). Yet, there are other ways to achieve this.

In these next three chapters, we explore another way to express anaphora. We follow Suppes (1976) and Böttner (1992) and use an extended relation algebra. This framework is natural for the variable-free Montagovian syntax and the BGQ logic, since it extends the quantifying structure by allowing the domain of quantification be a binary predicate. Hence, both the domain of quantification and the quantified predicate are binary relations (that is, sets which depend on a variable). Moreover, the output of such quantifying structures is also a binary relation, leading to an extended algebra of binary relations. These three chapters don't focus on inference, but rather on representing English in this framework.

To ease the notation, the idea of integrating the domain of quantification into the quantifier is temporarily rejected. In chapter 7, we reinstate the idea and show how to extend the BGQ logic to enclose the extended relation algebra.

Intuitively, a motivation for this type of representation, apart from curiosity and mathematical elegance, is to restrict the number of variables used, as natural language often doesn't need more than one. For practical purposes, the work presented here might be usable by a representation that takes advantage of the inference mechanism of the variable-free Montagovian syntax and other logics (namely, description logics), since its syntax encloses the others; see chapter 7.

From the perspective of the extended relation algebra framework (from now on, ERA), our main result is its extension to arbitrary determiners and the account of some English sentences involving long-distance anaphora (4.2), *'other'* (4.3), and donkeys (4.4).

(4.2) Every man_1 loves a woman who likes his_1 mother.

(4.3) Every man_1 loves the wife of $\textit{another}_1$ man.

(4.4) Every man who owns a donkey_1 beats it_1.

The chapters are divided as follows: in the current chapter, the semantics of ERA are presented, followed by various examples of English sentence represented in ERA; chapter 5 automates the semantic interpretation process; chapter 6 discusses the issues of quantification and anaphora in ERA, in particular, how set construction for the donkey sentences is handled.

4.2 Definition

We define our extended relation algebra as a boolean algebra of binary relations[1], augmented with some special operators, namely, the inverse operator and the composition operator, which serve as relation builders. (For convenience, we also define the domain-projection operator.)

Boolean algebra of binary relations Set of binary relations R ($R \subseteq \mathcal{U} \times \mathcal{U}$) and the operations of intersection, union, and complementation.

Inverse operator The inverse operator (4.5) permutes the two arguments of a relation.

Composition operator The composition operator (4.6) combines two relations to produce a new one. $R_1 \circ R_2$ *relates*[2] x to y if and only if there is a z such that R_1 relates x to z and R_2 relates z to y.

Domain-projection operator The domain-projection operator (4.7) computes the domain of a relation, that is, the set of elements that might appear on the left of the relation.

Formally, the operators are defined as follows:[3]

$$xR^{-1}y \triangleq yRx; \tag{4.5}$$

$$R_1 \circ R_2 \triangleq \{\,(x, y) \mid \exists z,\ xR_1z \wedge zR_2y\,\}; \tag{4.6}$$

$$\mathrm{dom}(R) \triangleq \{\,x \mid \exists y,\ xRy\,\}. \tag{4.7}$$

[1] In this chapter, if the arity of a relation is not mentioned it is a binary relation.
[2] We say that R relates x to y if and only if $(x, y) \in R$.
[3] Note that the following notations are equivalent: $xRy \leftrightarrow R(x, y) \leftrightarrow (x, y) \in R$.

Two useful constants are the identity relation I and the universal relation $\mathcal{U} \times \mathcal{U}$.

$$I \triangleq \{ (x, x) \mid x \in \mathcal{U} \}; \tag{4.8}$$

$$\mathcal{U} \times \mathcal{U} = \{ (x, y) \mid x \in \mathcal{U} \wedge y \in \mathcal{U} \}. \tag{4.9}$$

Note that the domain-projection operator is technically superfluous since it can be expressed in terms of the composition operator, but it is defined to improve readability.

$$\mathrm{dom}(R) \times \mathcal{U} = R \circ (\mathcal{U} \times \mathcal{U}). \tag{4.10}$$

For that same reason, we define a fourth operator, Φ, which will be needed to account for anaphora. We call it the *mirror* operator. It *aligns* the two arguments of a relation, that is, it computes the set of elements for which the relation is reflexive and, in order to get a binary relation, crosses the result with $\mathcal{U}$.

$$\Phi(R) \triangleq dom(R \cap I) \times \mathcal{U}. \tag{4.11}$$

4.3 Some English sentences

Before we start, here are the sets and relations used by our examples.

`john`, `mary`, `bill` are singleton sets, their unique element being the entity they denote.

`man`, `woman` are the sets of all men and all women, respectively.

`toy-of`, `friend-of`, `child-of`, `father-of`, `mother-of`, `wife-of` are relations with the obvious denotation. Note that the order of the arguments is like English; x `toy-of` y means that x is a toy of y.

`love`, `own`, `beat` are relations denoting verbs. Again, the order is like English, that is, the subject is at left, and the object is at right.

4.3.1 Simple sentences

The reader will note that each example is divided as follows: a sentence, its semantic representation just below, and then a step-by-step construction of the semantic representation. Each step has an English-like representation on the left and on the right, its analog in ERA.

Here, we should point out a difference between the overall representation of a sentence and its step-by-step representation, namely the use of the assignment operator. The overall representation is a functional representation (every relation is a function of the primary relations); yet, our use of the assignment operator guaranty that a functional representation is possible, since every assignment is of the type $R_1 = R_2 \circ R_3$ where R_2 and R_3 are also obtained by assignment or are constant.

There are many ways to express a simple sentence like (4.12) even in an extended relation algebra. Here, we propose an unconventional way that is justified in order to allow a uniform treatment with more complex noun phrases.

(4.12) John loves Mary.

$$\mathcal{U} \times \mathcal{U} = (\mathcal{U} \times \texttt{john}) \circ [\texttt{love} \circ (\texttt{mary} \times \mathcal{U})]$$

1. $xR_1y \leftrightarrow x$ loves y. $\qquad R_1 = \texttt{love}$
2. $xR_2y \leftrightarrow x$ loves Mary. $\qquad R_2 = R_1 \circ (\texttt{mary} \times \mathcal{U})$
3. John loves Mary. $\qquad \mathcal{U} \times \mathcal{U} = (\mathcal{U} \times \texttt{john}) \circ R_2$

In the analysis of sentence (4.12), the composition of `love` and $(\texttt{mary} \times \mathcal{U})$ produces R_2, the set of entities that *'love Mary'*. By definition, $\texttt{love} \circ (\texttt{mary} \times \mathcal{U})$ relates x to y if and only if there is a z such that x `love` z and $z \in \texttt{mary}$, that is, if and only if *'x loves Mary'*. (Note that y is not restricted here and can be anything.) Similarly, $(\mathcal{U} \times \texttt{john}) \circ R_2$ relates x to y if and only if there is a z such that $z \in \texttt{john}$ and *'z loves Mary'*, that is, if and only if *'John loves Mary'*. Moreover, because x and y are unrestricted, $(\mathcal{U} \times \texttt{john}) \circ R_2$ equals $\mathcal{U} \times \mathcal{U}$ if *'John loves Mary'* is true and $\emptyset$ otherwise. Thus, *'John loves Mary'* is asserted to be true or false by having this final relation equal to $\mathcal{U} \times \mathcal{U}$ or $\emptyset$, respectively.

(4.13) John loves at least one woman.

$$\mathcal{U} \times \mathcal{U} = (\mathcal{U} \times \texttt{john}) \circ [\texttt{love} \circ (\texttt{woman} \times \mathcal{U})]$$

1. $xR_1y \leftrightarrow x$ loves y. $\qquad R_1 = \texttt{love}$
2. $xR_2y \leftrightarrow x$ loves at least one woman. $\qquad R_2 = R_1 \circ (\texttt{woman} \times \mathcal{U})$
3. John loves at least one woman. $\qquad \mathcal{U} \times \mathcal{U} = (\mathcal{U} \times \texttt{john}) \circ R_2$

This variation of the previous sentence follows by replacing the set `mary` by `woman`. Again, $\texttt{love} \circ (\texttt{woman} \times \mathcal{U})$ relates x to y if and only if there is a z such that x `love` z and $z \in \texttt{woman}$, that is, if and only if *'x loves at least one woman'*. Therefore, R_2 equals $\mathcal{U} \times A$ where A is the set of entities that *'love at least one woman'*. The rest of the analysis is identical to the previous sentence.

4.3.2 Anaphora

The next examples deal with anaphora and range from the reflexive pronoun to the possessives. These are discussed by Böttner (1992), who attributes the original analysis to Suppes (1976). (Böttner actually extends the analysis to account for expressions containing the words *'each other'*, *'one another'*, *'the same'*, or *'a different'*.) Our treatment differs slightly, but is equivalent. The first example of anaphora is with a reflexive pronoun.

(4.14) John$_1$ loves himself$_1$.

$$\mathcal{U} \times \mathcal{U} = (\mathcal{U} \times \texttt{john}) \circ \Phi(\texttt{love})$$

1. $xR_1y \leftrightarrow x$ loves y. $\quad R_1 = \texttt{love}$
2. $xR_2y \leftrightarrow x_1$ loves itself$_1$. $\quad R_2 = \Phi(R_1)$
3. John$_1$ loves himself$_1$. $\quad \mathcal{U} \times \mathcal{U} = (\mathcal{U} \times \texttt{john}) \circ R_2$

To represent this sentence, we use the mirror operator defined earlier. The set of entities that *'love itself'* is computed by applying the mirror operator to the relation `love`. The result is of the type $A \times \mathcal{U}$, where A is the sets of elements for which `love` is reflexive, that is, the set of entities that *'love itself'*.

(4.15) John$_1$ loves at least one of his$_1$ toys.

$$\mathcal{U} \times \mathcal{U} = (\mathcal{U} \times \texttt{john}) \circ \Phi(\texttt{love} \circ \texttt{toy-of})$$

1. $xR_1y \leftrightarrow x$ loves y. $\quad R_1 = \texttt{love}$
2. $xR_2y \leftrightarrow x$ loves at least one toy of y. $\quad R_2 = R_1 \circ \texttt{toy-of}$
3. $xR_3y \leftrightarrow x_1$ loves at least one of its$_1$ toys. $\quad R_3 = \Phi(R_2)$
4. John$_1$ loves at least one of his$_1$ toys. $\quad \mathcal{U} \times \mathcal{U} = (\mathcal{U} \times \texttt{john}) \circ R_3$

The example (4.15) uses the same idea except that the mirror operator is not applied to the verb directly, but to the composition of `love` and `toy-of`, which relates x to y if and only if there is a z such that x `love` z and z `toy-of` y, that is, if and only if *'x loves at least one toy of y'*. The result of the alignment is the set of entities that *'love at least one of its toys'*.

(4.16) John$_1$ loves at least one toy of at least one of his$_1$ friends.

$$\mathcal{U} \times \mathcal{U} = (\mathcal{U} \times \texttt{john}) \circ \Phi(\texttt{love} \circ \texttt{toy-of} \circ \texttt{friend-of})$$

1. $xR_1y \leftrightarrow x$ loves y. $\quad R_1 = \texttt{love}$
2. $xR_2y \leftrightarrow x$ loves at least one toy of y. $\quad R_2 = R_1 \circ \texttt{toy-of}$
3. $xR_3y \leftrightarrow x$ loves at least one toy of at least one friend of y. $\quad R_3 = R_2 \circ \texttt{friend-of}$
4. $xR_4y \leftrightarrow x_1$ loves at least one toy of at least one of its$_1$ friends. $\quad R_4 = \Phi(R_3)$
5. John$_1$ loves at least one toy of at least one of his$_1$ friends. $\quad \mathcal{U} \times \mathcal{U} = (\mathcal{U} \times \texttt{john}) \circ R_4$

As this example shows, the composition operator can be applied more than once before aligning the relation.

This completes for now the account of English sentences using the four operators defined at the beginning. The reader will note that the examples are restricted to the determiner *'at least one'*. It is possible to account for *'every'* and *'no'*, but the notation becomes harder to follow. It is simpler to define new operators to keep the notation close to English; we do this in the next section.

4.4 New family of composition operators

The problem with expressing the meaning of natural language sentences only in terms of the given operators is that the notation becomes obscure for *'no'* and *'every'* since they are obtained by negation and double negation respectively. For instance, sentences like (4.17) and (4.18) must be paraphrased using negation (as shown in parentheses). Moreover, not every determiner can be expressed in terms of the four given operators.[4] This justifies the need for a new class of operators.

(4.17) John loves nobody. (It is not true that John loves somebody.)

(4.18) John loves everybody. (It is not true that John doesn't love somebody.)

In order to handle such determiners with the same direct correspondence to English as for *'at least one'*, we propose a family of composition operators, one representing each determiner. The operators are defined with respect to Moore's notation (Moore, 1981) presented in chapter 2. Note that D can be any determiner from *'some'*, *'every'*, *'no'*, *'at least k'*, *'at most k'*, *'exactly k'*, *'most'*, etc.

$$R_1 \circ_D R_2 \;\triangleq\; \{\,(x,y) \mid (D; z; R_2(z,y); R_1(x,z))\,\}. \tag{4.19}$$

The original composition operator corresponds to the operator defined with *'at least one'*.

$$R_1 \circ R_2 \;=\; R_1 \circ_{\text{at-least-1}} R_2. \tag{4.20}$$

The equality follows by definition since the determiner *'at least one'* in Moore's syntax corresponds to the existential quantifier.

$$\begin{aligned}
(\mathbf{atleast}_1; z; R_2(z,y); R_1(x,z)) &\iff |\{\,z \mid xR_1z\,\} \cap \{\,z \mid zR_2y\,\}| \geq 1;\\
&\iff \exists z,\; xR_1z \wedge zR_2y;\\
&\iff x(R_1 \circ R_2)y.
\end{aligned} \tag{4.21}$$

[4]For sure, *'most'* cannot be expressed (at least in the functional representation); but even *'at least two'* cannot (contrary to first-order logic, a variable cannot hold a binding in order to bind another variable with a different value).

4.4.1 Object position

The operators are defined in such a way that $R_1 \circ_D R_2$ literally means in English 'R_1 D R_2' if R_1 denotes a verb and R_2 a common noun; that is, $\texttt{love} \circ_{\text{most}} \texttt{toy-of}$ relates x to y if and only if '*x loves most toys of y*', and so on for every determiner. This follows by definition, because $\texttt{love} \circ_D \texttt{toy-of}$ relates x to y if and only if $(D; z; z\ \texttt{toy-of}\ y; x\ \texttt{love}\ z)$, that is, if for any given x and y, D describes the relationship between the set of '*toys of y*' and the set of entities '*loved by x*'.

(4.22) $xRy \leftrightarrow x$ loves D toy(s) of y. $R = \texttt{love} \circ_D \texttt{toy-of}$

Note that the correspondence still holds if the second relation is not a relation noun like $\texttt{toy-of}$, that is, $R \circ_D (S \times \mathcal{U})$ literally means 'R D S' if R denotes a verb, and S a common noun. For instance, $\texttt{love} \circ_{\text{most}} (\texttt{woman} \times \mathcal{U})$ relates x to y if and only if '*x loves most women*'. This follows since in $\texttt{love} \circ_D (\texttt{woman} \times \mathcal{U})$ corresponds to $(D; z; \texttt{woman}(z); x\ \texttt{love}\ z)$, which is true if and only if for any given x and y, D describes the relationship between the set of '*women*' and the set of entities '*loved by x*'.

(4.23) $xRy \leftrightarrow x$ loves D woman (women). $R = \texttt{love} \circ_D (\texttt{woman} \times \mathcal{U})$

4.4.2 Subject position

The composition operators can also be used by subject NP's, but since they are not, in general, symmetric ($[R_1 \circ_D R_2]^{-1}$ does not necessarily equal ${R_2}^{-1} \circ_D {R_1}^{-1}$), the order must be reversed compared to English, that is, the relation representing the VP must be on the left of the operator, and the relation representing the NP on the right.

(4.24) Most men love some women.

$$\mathcal{U} \times \mathcal{U} = [\texttt{love} \circ_{\text{some}} (\texttt{woman} \times \mathcal{U})]^{-1} \circ_{\text{most}} (\texttt{man} \times \mathcal{U})$$

1. $xR_1y \leftrightarrow x$ loves y. $R_1 = \texttt{love}$
2. $xR_2y \leftrightarrow x$ loves some women. $R_2 = R_1 \circ_{\text{some}} (\texttt{woman} \times \mathcal{U})$
3. Most men love some women. $\mathcal{U} \times \mathcal{U} = {R_2}^{-1} \circ_{\text{most}} (\texttt{man} \times \mathcal{U})$

The sentence is true if '*most*' describes the relationship between the set of '*men*' and the set of entities that '*love some women*'. (R_2 is reversed, so the subject is on the right of the relation and can be quantified using a composition operator.)

4.4.3 More composition operators for English

The focus of ERA is on the study of the dependencies, but ERA is far from covering all the subtleties of natural language. In particular, ERA lacks the ability to represent presuppositions, and this is crucial to represent definites. So, we are confined to coarse approximations of the meaning, which we try to improve by defining these new operators.

Two operators are defined for the determiner *'some'*: $\circ_{\texttt{some1}}$ for the singular case, and $\circ_{\texttt{some}}$ for the plural case. This captures the distinction made by natural language that the plural version requires *'at least two'* objects.

$$P \circ_{\texttt{some1}} R \triangleq P \circ_{\texttt{at-least-1}} R; \tag{4.25}$$

$$P \circ_{\texttt{some}} R \triangleq P \circ_{\texttt{at-least-2}} R. \tag{4.26}$$

Definites are a major problem for ERA, because they require presuppositions. The presupposition is modeled as an additional statement, which, in the case of the singular *'the'*, requires the quantification domain to have exactly one element, or, for the plural case, requires it to have at least two elements. If the presuppositions are not respected, the statement is false, which is another approximation since the non-respect of a presupposition is not the same type of falsehood than a false main clause. Moreover, this patch does not hold under negation because the presupposition would be negated, but should not be. Yet, it will do for our examples.

$$P \circ_{\texttt{the1}} R \triangleq (P \circ_{\texttt{every}} R) \cap [(\mathcal{U} \times \mathcal{U}) \circ_{\texttt{exactly-1}} R]; \tag{4.27}$$

$$P \circ_{\texttt{the}} R \triangleq (P \circ_{\texttt{every}} R) \cap [(\mathcal{U} \times \mathcal{U}) \circ_{\texttt{at-least-2}} R]. \tag{4.28}$$

4.5 More English sentences

Before we continue with the semantic interpretation (the next chapter), we show examples of English sentences encoded using the relation operations, which cover the syntactic constructs handled by the semantic interpretation process of the next chapter.

4.5.1 'Of' prepositional phrases

The idea for handling possessives and *'of'* prepositional phrases is essentially the same. Basically, the common noun of the main NP, instead of denoting a class, say `child`, denotes a relation, `child-of`, which is a class depending on a variable. Consequently, the main NP before the action of its *'of'* prepositional phrase replaces the variable it is quantifying by another dependency, which is to be quantified by the prepositional phrase.

(4.29) A child of every man loves Mary.

$$\mathcal{U} \times \mathcal{U} = \{[\texttt{love} \circ_{\texttt{the1}} (\texttt{mary} \times \mathcal{U})]^{-1} \circ_{\texttt{some1}} \texttt{child-of}\} \circ_{\texttt{every}} (\texttt{man} \times \mathcal{U})$$

1. $xR_1y \leftrightarrow x$ loves y. $\qquad R_1 = \texttt{love}$

2. $xR_2y \leftrightarrow x$ loves Mary. $R_2 = R_1 \circ_{\mathtt{the1}} (\mathtt{mary} \times \mathcal{U})$

3. $xR_3y \leftrightarrow$ A child of y loves Mary. $R_3 = {R_2}^{-1} \circ_{\mathtt{some1}} \mathtt{child\text{-}of}$

4. A child of every man loves Mary. $\mathcal{U} \times \mathcal{U} = R_3 \circ_{\mathtt{every}} (\mathtt{man} \times \mathcal{U})$

In the example, the partial NP *'a child of'* maps the VP *'loves Mary'* to the partial sentence *'a child of y loves Mary'*; then y is quantified away by *'every man'*.

4.5.2 Relative clauses

We continue with examples of relative clauses. A relative clause restricts the common noun of an NP. For instance, in the following example (4.30), the relative clause restricts the set of *'women'* to those *'whom John loves* gap*'*.

(4.30) x (is a) woman John loves *gap*.

$$(\mathtt{woman} \times \mathcal{U}) \cap [\mathtt{love}^{-1} \circ_{\mathtt{the1}} (\mathtt{john} \times \mathcal{U})]^{-1}$$

1. $xR_1y \leftrightarrow x$ loves *gap* y. $R_1 = \mathtt{love}$

2. $xR_2y \leftrightarrow$ John loves *gap* x. $R_2 = {R_1}^{-1} \circ_{\mathtt{the1}} (\mathtt{john} \times \mathcal{U})$

3. $xR_3y \leftrightarrow x$ (is a) woman John loves *gap*. $R_3 = (\mathtt{woman} \times \mathcal{U}) \cap R_2$

A relative clause is basically a sentence missing an NP (the *gap*). The set it denotes are the entities which could fill the gap and make the sentence true. For instance, the set of entities *'John loves'* is the set of every entity d such that *'John loves d'*. In ERA, this set is obtained by treating the relative clause as any sentence, except for the gap which maps the incoming relation to itself and whose inaction leaves the incoming relation unquantified on that argument. For instance, in example (4.30), the gap leaves the relation `love` intact; then, the integration of *'John'* (as the subject of the relative clause) creates the set of entities *'John loves'* since the resulting relation (R_2) relates x to y if and only if *'John'* is part of the things that *'love x'* (y is left free).

The next example differs in two ways with the one just seen: first, the gap is not in object position; second, *'whose'* is used to introduce the relative clause.

(4.31) x (is a) woman whose mother loves John. (x woman the mother of whom (*gap*) loves John.)

$$(\mathtt{woman} \times \mathcal{U}) \cap \{[\mathtt{love} \circ_{\mathtt{the1}} (\mathtt{john} \times \mathcal{U})]^{-1} \circ_{\mathtt{the1}} \mathtt{mother\text{-}of}\}^{-1}$$

1. $xR_1y \leftrightarrow x$ loves John. $R_1 = \mathtt{love} \circ_{\mathtt{the1}} (\mathtt{john} \times \mathcal{U})$

2. $xR_2y \leftrightarrow$ *gap* y's mother loves John. $R_2 = {R_1}^{-1} \circ_{\mathtt{the1}} \mathtt{mother\text{-}of}$

3. $xR_3y \leftrightarrow x$ (is a) woman whose mother *gap* loves John. $R_3 = (\mathtt{woman} \times \mathcal{U}) \cap {R_2}^{-1}$

A gap on the subject side is handled as one on the object side. However, the final relation has to be reversed, so the left side holds the set of entities satisfying the relative clause. In the example above, *'whose'* is accounted by paraphrasing *'whose* NP$''$*'* by *'the* NP$''$ *of whom'* (an NP$''$ is an NP before the action of its prepositional phrase or possessor). The relative clause is treated as a normal sentence except for the gap which denotes the identity function.

Finally, we give an example of a relative clause in a complete sentence. (R_3 is taken from the previous example.)

(4.32) Bill loves a woman whose mother *gap* loves John.

1. $xR_4y \leftrightarrow x$ loves a woman whose mother *gap* loves John. $\qquad R_4 = \texttt{love} \circ_{\text{some1}} R_3$
2. Bill loves a woman whose mother *gap* loves John. $\qquad \mathcal{U} \times \mathcal{U} = R_4^{-1} \circ_{\text{the1}} (\texttt{bill} \times \mathcal{U})$

4.5.3 Verb *'to be'* and *'to have'*

The verbs *'to be'* and *'to have'* have semantics different from the verbs in the examples seen.

To be The verb *'to be'* has two different usages which can be handled directly.

(4.33) Every man is bald.

$$(\mathcal{U} \times \texttt{bald}) \circ_{\text{every}} (\texttt{man} \times \mathcal{U})$$

(4.34) Every man is a mammal.

$$[\texttt{I} \circ_{\text{some1}} (\texttt{mammal} \times \mathcal{U})]^{-1} \circ_{\text{every}} (\texttt{man} \times \mathcal{U})$$

In the first case, the adjective predicates the subject. In the second case, the identity relation is taken as the verb.

To have The verb *'to have'* has three usages covered here. The distinction between the two first usages is subtle. Consider the following sentence.

(4.35) John has every car.

1. $xR_1y \leftrightarrow y$ has every car. $\qquad R_1 = \texttt{possess} \circ_{\text{every}} (\texttt{car} \times \mathcal{U})$
2. John has every car. $\qquad \mathcal{U} \times \mathcal{U} = R_1^{-1} \circ_{\text{the1}} (\texttt{john} \times \mathcal{U})$

In the first case, alienable possession, the verb *'to have'* is understood as any other verb. (Its denotation is the `possess` binary relation.)

In the second case, inalienable possession, the verb *'to have'* has a structural meaning; it indicates a special way to combine the subject and complement, that is, the sentence 'X has Y' is understood as 'Y of X is' (where *'is'* is the existential predicate of Descartes!, or put simply the universal predicate). Obviously, this reading is only possible if the object NP depends on a free variable, which is so if the common noun is a genuine relation like `child-of` (and the free variable is not bound by another mean).

(4.36) John has two children. (Two children of John are.)

1. $xR_1y \leftrightarrow y$ has two children. $R_1 = (\mathcal{U} \times \mathcal{U}) \circ_{\text{at-least-2}} \texttt{child-of}$

In this example, $\mathcal{U} \times \mathcal{U} \circ_{\text{at-least-2}} \texttt{child-of}$ relates x to y if and only if the set of z such that *'z child of y'* contains *'at least two'* elements in $\mathcal{U}$, that is, if and only if *'y has at least two children'*, as intended.

(4.37) John has no children. (No children of John are.)

1. $xR_1y \leftrightarrow y$ has no children. $R_1 = (\mathcal{U} \times \mathcal{U}) \circ_{\text{no}} \texttt{child-of}$

In this other example, $(\mathcal{U} \times \mathcal{U}) \circ_{\text{no}} \texttt{child-of}$ relates x to y if and only if *'no'* element of the set of z such that *'z child of y'* is in $\mathcal{U}$, which is possible if and only if *'y has no children'*. As with English, this reading is available only for the weak determiners (in the sense of Barwise and Cooper (1981)) as it would be a tautology for strong determiners like *'every'* or *'most'*. For instance, the following sentence can only be understood in the alienable sense, since otherwise, it means that *'most children of John'* are in $\mathcal{U}$, which is true of *'every child of John'* however many *'children John has'*.

(4.38) John has most children.

Notice that the object dependency to a free variable can also be introduced by an anaphor in the relative clause, as in the following example.

(4.39) Every woman$_1$ has a man who loves her$_1$. (For every woman$_1$, a man who loves her$_1$ is.)

1. xR_1y$\leftrightarrow$ x man who loves y. $R_1 = (\texttt{man} \times \mathcal{U}) \cap \texttt{love}$
2. xR_2y$\leftrightarrow$ y_1 has a man who loves it$_1$. $R_2 = (\mathcal{U} \times \mathcal{U}) \circ_{\text{some1}} R_1$
3. Every woman$_1$ has a man who loves her$_1$. $\mathcal{U} \times \mathcal{U} = R_2 \circ_{\text{every}} (\texttt{woman} \times \mathcal{U})$

There is a third usage of the verb *'to have'*. Then, as well, *'have'* has a structural meaning; the verb phrase is interpreted as an ordinary one, except that the binary predicate is not denoted by the verb, but by the relation appearing after the word *'as'*.

(4.40) John has Mary as a friend. (John stands in a friend relationship with Mary.)

1. $xR_1y \leftrightarrow x$ friend-of Mary. $R_1 = \texttt{friend-of}^{-1} \circ_{\text{the1}} (\texttt{mary} \times \mathcal{U})$
2. John has Mary as a friend. $\mathcal{U} \times \mathcal{U} = R_1^{-1} \circ_{\text{the1}} (\texttt{john} \times \mathcal{U})$

4.5.4 Long-distance anaphora

So far, the examples of anaphora are always direct, the complement (or complement of the complement, etc) referring to the subject. The next two pairs of examples show one case of anaphora where the referent is the noun complement of the subject and one case where the referent is outside the relative clause in which the anaphor occurs. Moreover, the examples are grouped in pairs that emphasize the differences caused by the selection of the referent for the possessive adjective.

Referent is a noun complement of the subject

In this next example (4.41), the referent of the possessive adjective is the subject, while it is the noun complement of the subject in example (4.42).

(4.41) Every man's wife$_1$ loves her$_1$ mother.

$$\mathcal{U} \times \mathcal{U} = \Phi(\texttt{love} \circ_{\texttt{the1}} \texttt{mother-of})^{-1} \circ_{\texttt{the1}} \texttt{wife-of} \circ_{\text{every}} (\texttt{man} \times \mathcal{U})$$

1. $xR_1y \leftrightarrow x$ loves y. $\qquad R_1 = \texttt{love}$
2. $xR_2y \leftrightarrow x$ loves the mother of y. $\qquad R_2 = R_1 \circ_{\texttt{the1}} \texttt{mother-of}$
3. $xR'_2y \leftrightarrow x_1$ loves its$_1$ mother. $\qquad R'_2 = \Phi(R_2)$
4. $xR_3y \leftrightarrow y$'s wife$_1$ loves her$_1$ mother. $\qquad R_3 = R_2'^{-1} \circ_{\texttt{the1}} \texttt{wife-of}$
5. Every man's wife$_1$ loves her$_1$ mother. $\qquad \mathcal{U} \times \mathcal{U} = R_3 \circ_{\text{every}} (\texttt{man} \times \mathcal{U})$

(4.42) Every man$_1$'s wife loves his$_1$ mother.

$$\mathcal{U} \times \mathcal{U} = \Phi(\{[\texttt{love} \circ_{\texttt{the1}} \texttt{mother-of}]^{-1} \circ_{\texttt{the1}} \texttt{wife-of}\})^{-1} \circ_{\text{every}} (\texttt{man} \times \mathcal{U})$$

1. $xR_1y \leftrightarrow x$ loves y. $\qquad R_1 = \texttt{love}$
2. $xR_2y \leftrightarrow x$ loves the mother of y. $\qquad R_2 = R_1 \circ_{\texttt{the1}} \texttt{mother-of}$
3. $xR_3y \leftrightarrow y$'s wife loves the mother of x. $\qquad R_3 = R_2^{-1} \circ_{\texttt{the1}} \texttt{wife-of}$
4. $xR'_3y \leftrightarrow y_1$'s wife loves its$_1$ mother. $\qquad R'_3 = \Phi(R_3)^{-1}$
5. Every man$_1$'s wife loves his$_1$ mother. $\qquad \mathcal{U} \times \mathcal{U} = R'_3 \circ_{\text{every}} (\texttt{man} \times \mathcal{U})$

The representation of the two sentences differs in the relation to which the mirror operator is applied. In the first case, applying the mirror operator to the relation *'x loves the mother of y'* produces the relation (x_1 loves its$_1$ mother), which is mapped to *'y's wife$_1$ loves her$_1$ mother'*. In the second case, not applying it leaves *'x loves the mother of y'* intact, so the integration of `wife-of` produces *'y's wife loves the mother of x'* instead; then, the application of the mirror operator at this point binds x to the noun complement of the subject by producing the relation *'y_1's wife loves its$_1$ mother'*.

In general, this is how we will account for anaphora: the pronoun leaves unbound the side of the relation it is quantifying, yet the application of the mirror operator just before the integration of the referent has the effect of binding the pronoun to the referent. Note that this mechanism requires the free variable to exist in the scope of the referent, that is, the referent must be integrated with the meaning after the anaphor (pronoun or possessive adjective). The free variable must also somehow survive until then. This latter condition is not guaranteed, since encoding the free variable freezes one side of the relation, which might be required by other constructs, like another anaphor. We call this the one-variable constraint.

Referent is outside the relative clause

The mechanism just described works as well if the anaphor (pronoun or possessive adjective) is inside a relative clause, and the referent outside it. Consider the two following examples:

(4.43) Every man loves a woman$_1$ who loves her$_1$ mother.

$$\mathcal{U} \times \mathcal{U} = \{\texttt{love} \circ_{\text{some1}} [(\mathcal{U} \times \texttt{woman}) \cap \Phi(\texttt{love} \circ_{\text{the1}} \texttt{mother-of})]\}^{-1} \circ_{\text{every}} (\texttt{man} \times \mathcal{U})$$

1. $xR_1y \leftrightarrow x$ loves y. $\quad R_1 = \texttt{love}$
2. $xR_2y \leftrightarrow x$ loves the mother of y. $\quad R_2 = R_1 \circ_{\text{the1}} \texttt{mother-of}$
3. $xR'_2y \leftrightarrow x_1$ loves its$_1$ mother. $\quad R'_2 = \Phi(R_2)$
4. $xR_3y \leftrightarrow x$ loves a woman$_1$ who loves her$_1$ mother. $R_3 = \texttt{love} \circ_{\text{some1}} [(\texttt{woman} \times \mathcal{U}) \cap R'_2]$
5. Every man loves a woman$_1$ who loves her$_1$ mother. $\quad \mathcal{U} \times \mathcal{U} = R_3^{-1} \circ_{\text{every}} (\texttt{man} \times \mathcal{U})$

(4.44) Every man$_1$ loves a woman who loves his$_1$ mother.

$$\mathcal{U} \times \mathcal{U} = \Phi(\{\texttt{love} \circ_{\text{some1}} [(\mathcal{U} \times \texttt{woman}) \cap (\texttt{love} \circ_{\text{the1}} \texttt{mother-of})]\})^{-1} \circ_{\text{every}} (\texttt{man} \times \mathcal{U})$$

1. $xR_1y \leftrightarrow x$ loves y. $\quad R_1 = \texttt{love}$
2. $xR_2y \leftrightarrow x$ loves y's mother. $\quad R_2 = R_1 \circ_{\text{the1}} \texttt{mother-of}$
3. $xR_3y \leftrightarrow x$ loves a woman who loves y's mother. $\quad R_3 = \texttt{love} \circ_{\text{some1}} [(\texttt{woman} \times \mathcal{U}) \cap R_2]$
4. $xR'_3y \leftrightarrow x_1$ loves a woman who loves its$_1$ mother. $\quad R'_3 = \Phi(R_3)$
5. Every man$_1$ loves a woman who loves his$_1$ mother. $\quad \mathcal{U} \times \mathcal{U} = R_3'^{-1} \circ_{\text{every}} (\texttt{man} \times \mathcal{U})$

In the first sentence, the possessive adjective refers to the subject of the relative clause; while in the second one, the possessive adjective refers to the subject of the whole sentence (*'every man'*). Yet, the two examples differ only on the referent of the possessive adjective (and on its gender), which is reflected in the semantic representation by the application of the mirror operator at different stages.

The immediate application of the mirror operator on the relation denoted by the relative clause produces the set of *'x_1 who loves its$_1$ mother'* ($\Phi(\texttt{love} \circ_{\text{the1}} \texttt{mother-of})$); while its non-application leaves the relation *'x who loves the mother of y'* intact (which can be viewed as the set, depending on y, of x's such that *'x who loves the mother of y'*) ($\texttt{love} \circ_{\text{the1}} \texttt{mother-of}$). So, the restriction relation $R(x,y)$ for the object NP is *'x (is a) woman$_1$ who loves her$_1$ mother'* and *'x (is a) woman who loves the mother of y'*, respectively. From that point, the latter case is similar to a possessive referring to the subject by applying the mirror operator just before the integration of the subject.

4.5.5 *'Other'*

Before moving on to more complex cases of anaphora, we show how to represent the word *'other'* in this formalism.

(4.45) John loves Mary$_1$. Bill loves another$_1$ woman.

$$\mathcal{U} \times \mathcal{U} = \{\texttt{love} \circ_{\text{some1}} [(\texttt{woman} \times \mathcal{U}) \cap \overline{(\texttt{mary} \times \mathcal{U})}]\}^{-1} \circ_{\text{some1}} (\texttt{bill} \times \mathcal{U})$$

1. $xR_1y \leftrightarrow x$ loves y. $\qquad R_1 = \texttt{love}$
2. $xR_2y \leftrightarrow x$ loves a woman other than Mary. $R_2 = R_1 \circ_{\text{some1}} [(\texttt{woman} \times \mathcal{U}) \cap \overline{(\texttt{mary} \times \mathcal{U})}]$
3. Bill loves a woman other than Mary. $\qquad \mathcal{U} \times \mathcal{U} = R_2^{-1} \circ_{\text{the1}} (\texttt{bill} \times \mathcal{U})$

In this example, *'Mary'* is the referent of *'another'* (analyzed as *'an'* + *'other'*), which excludes her from the restriction relation of the NP ($[(\texttt{woman} \times \mathcal{U}) \cap \overline{(\texttt{mary} \times \mathcal{U})}]$), that is, *'another woman'* is understood as *'a woman other than Mary'*.

A similar analysis works also when the referent outscopes *'other'*. Consider the following example.

(4.46) Some men$_1$ love the wife of another$_1$ man.

$$\mathcal{U} \times \mathcal{U} = \Phi(\{[\texttt{love} \circ_{\text{the1}} \texttt{wife-of}] \circ_{\text{some}} [(\texttt{man} \times \mathcal{U}) \cap \bar{I}]\})^{-1} \circ_{\text{some1}} (\texttt{man} \times \mathcal{U})$$

1. $xR_1y \leftrightarrow x$ loves y. $\qquad R_1 = \texttt{love}$
2. $xR_2y \leftrightarrow x$ loves the wife of y. $\qquad R_2 = R_1 \circ_{\text{the1}} \texttt{wife-of}$
3. $xR_3y \leftrightarrow x$ loves the wife of a man other than y. $\qquad R_3 = R_2 \circ_{\text{some1}} [(\texttt{man} \times \mathcal{U}) \cap \bar{I}]$
4. $xR_4y \leftrightarrow x_1$ loves the wife of another$_1$ man. $\qquad R_4 = \Phi(R_3)$
5. Some men$_1$ love the wife of another$_1$ man. $\qquad \mathcal{U} \times \mathcal{U} = R_4^{-1} \circ_{\text{some1}} (\texttt{man} \times \mathcal{U})$

In that case, *'another man'* is understood as *'a man other than y'* (the restriction relation is $[(\texttt{man} \times \mathcal{U}) \cap \bar{I}]$), which y is eventually bound to the subject as done for a possessive.

More complex examples involving *'other'* discussed by Böttner (1992) are also expressible in this formalism, but they won't be part of the semantic interpretation process of the next chapter.

(4.47) They$_1$ love each other$_1$. (Each one$_1$ of them$_2$ loves the other$_1$ ones$_2$.)

1. $xR_1y \leftrightarrow x$ loves the ones other then y. $\qquad R_1 = \texttt{love} \circ_{\text{the}} [(\texttt{they} \times \mathcal{U}) \cap \bar{I}]$
2. $xR_2y \leftrightarrow x_1$ loves the other$_1$ ones. $\qquad R_2 = \Phi(R_1)$

3. They$_1$ love each other$_1$. $\mathcal{U} \times \mathcal{U} = R_2^{-1} \circ_{\texttt{the}} (\texttt{they} \times \mathcal{U})$

Chapter 5

Semantic Interpretation in ERA

In this chapter, we show how to compute the semantic representation of a sentence in ERA given the syntactic representation of the sentence. This process of computing the semantics from the syntax is called *semantic interpretation*. To do so, the semantic representation is computed recursively on the syntax as originally done by Montague (1974):

- Syntactic rules recursively define the structure of the sentences;
- Semantic rules construct semantic objects by combining them;
- Each syntactic structure in a sentence corresponds to a semantic object;
- The semantic representation is built in parallel with the syntax by having each semantic rule coupled to a syntactic rule.

However, we make an extensive use of under-specifications to ease the presentation and abstract away problems not of concern here.

5.1 Overview of the semantic interpretation process

5.1.1 Simplifications

Many simplifications are made, since we wish to keep the interpretation process simple.

- The root forms of the words are the terminal symbols for the grammar, and any inflection is ignored. (*'man'* and *'men'* input the same symbol, `man`, to the grammar.) Consequently, no kind of grammatical agreement, such as subject–verb agreement or object–subject case on pronouns, is handled.
- The string of symbols the grammar receives is slightly preprocessed for uniform treatment. For instance, the word *'another'* is broken into `an` followed by `other` for the grammar; also,

sentences with negation are reordered such that the negation operator appears in front of the verb, that is, *'is not'*, *'has not'*, and *'doesn't love'* are interpreted as *'not is'*, *'not has'*, and *'not loves'*, respectively.

- The grammar is under-specified to make it simpler. For instance, the gap-filling problem[1] is by-passed by having an NP derived from nothing ($\perp$) without any mechanism governing when such a derivation is valid.

- The semantics too are under-specified by having two potential semantic rules for a single syntactic rule. The motivation is to abstract away some mechanisms. For instance, the decision of applying the mirror operator to bind a variable to an NP is avoided by making the two choices possible.

- The mechanism for finding the referent of an anaphor is undefined.

- The scope given to the NP's facilitates the semantic interpretation.

5.1.2 Hooks

In order to make it possible to combine semantic objects, each semantic object has a number of *hooks* (the number depends on the syntactic type, NP or VP). A hook is a binary relation that the semantic object uses or produces. For example:

(5.1) John loves a woman.

1. $xR_1y \leftrightarrow x$ loves y. $\qquad R_1 = \texttt{love}$
2. $xR_2y \leftrightarrow x$ loves a woman. $\qquad R_2 = R_1 \circ_{\texttt{some}} (\texttt{woman} \times \mathcal{U})$
3. John loves a woman. $\qquad \mathcal{U} \times \mathcal{U} = R_2^{-1} \circ_{\texttt{the1}} (\texttt{john} \times \mathcal{U})$

In this example, the verb (V) and the verb phrase (VP) have each a single hook, V_d equal to R_1 and VP_d equal to R_2, corresponding to their respective denotation. An NP has two hooks: the relation it quantifies ($\mathrm{NP}_{\text{before}}$) and the resulting relation ($\mathrm{NP}_{\text{after}}$); we will say that a given NP maps its $\mathrm{NP}_{\text{before}}$ to its $\mathrm{NP}_{\text{after}}$. For instance, the NP *'a woman'* maps R_1 to R_2, and *'John'* maps R_2^{-1} to $\mathcal{U} \times \mathcal{U}$. Given this, the semantic rules operate by imposing constraints on the hooks. In this example, the semantics of the VP are built by adding two constraints: one that constrains $\mathrm{NP}_{\text{before}}$ of *'a woman'* and V_d of *'loves'* to the same value, and one that constrains $\mathrm{NP}_{\text{after}}$ of *'a woman'* and VP_d^{-1} of *'loves a woman'* to the same value. The result is that the NP *'a woman'* maps *'loves'* to the verb phrase.

[1] "The woman John loves *gap* is Mary".

5.1.3 Notation

The notation for the syntactic and semantic rules proceeds as follows:

- Each rule has a syntactic part on the left and its semantic counterpart on the right;
- The syntactic part is in BNF;
- The semantic part is a set of constraints on the hooks;
- The hooks are represented as follows: for instance, $\mathrm{NP}_{\mathrm{before}}$ is the hook named *'before'* of the semantic object corresponding to the NP of the syntactic part. Overall, there are six hooks: X_d, X_r, X_{in}, $\mathrm{X}_{\mathrm{out}}$, $\mathrm{X}_{\mathrm{before}}$, and $\mathrm{X}_{\mathrm{after}}$. They are named respectively: denotation, restriction, *'in'*, *'out'*, *'before'*, and *'after'*. Each semantic type has a subset of these hooks: most of them, S^*, S', VP, ADJ, CN, REL, N, and R, have only one denotation hook; POSS uses $\mathrm{POSS}_{\mathrm{in}}$ and $\mathrm{POSS}_{\mathrm{out}}$; the semantic objects of the type NP^* use all the hooks but the denotation one.

To shorten the semantic rules, a semantic object automatically inherits the hooks of the semantic objects from which it was built, unless otherwise specified by the semantic rule. For example, rule 28 of the grammar is understood as its alternate version 28′. (Note that a syntactic rule may have no specified semantic counterpart if the inheritance is sufficient to capture its semantics.)

(28) $\mathrm{NP}^{(3)} \leftarrow \mathrm{NP}^{(4)}$ REL $\qquad \mathrm{NP}_r^{(3)} = \mathrm{NP}_r^{(4)} \cap \mathrm{REL}_d$

(28′) $\mathrm{NP}^{(3)} \leftarrow \mathrm{NP}^{(4)}$ REL $\qquad \begin{cases} \mathrm{NP}_r^{(3)} = \mathrm{NP}_r^{(4)} \cap \mathrm{REL}_d \\ \mathrm{NP}_{\mathrm{before}}^{(3)} = \mathrm{NP}_{\mathrm{before}}^{(4)},\ \mathrm{NP}_{\mathrm{after}}^{(3)} = \mathrm{NP}_{\mathrm{after}}^{(4)} \\ \mathrm{NP}_{\mathrm{in}}^{(3)} = \mathrm{NP}_{\mathrm{in}}^{(4)},\ \mathrm{NP}_{\mathrm{out}}^{(3)} = \mathrm{NP}_{\mathrm{out}}^{(4)} \end{cases}$

5.2 Detailed view of the interpretation process

We proceed with a rule-by-rule view of the interpretation process.

5.2.1 Sentences

Complete sentences: rules 1 to 3

(1) $\mathrm{S} \leftarrow \mathrm{S}'$ $\qquad \mathrm{S}'_d = \mathcal{U} \times \mathcal{U}$

(2) S ← IT–IS–TRUE S′ $\qquad \mathrm{S}'_d = \mathcal{U} \times \mathcal{U}$

(3) S ← IT–IS–NOT–TRUE S′ $\qquad \mathrm{S}'_d = \emptyset$

Complete sentences (S) are complete semantic objects in the sense that they cannot be combined to other objects. No hooks are defined for them. The grammar covers three ways to derive a complete sentence: directly from an *unbound sentence*, or, from an unbound sentence preceded by IT–IS–TRUE or IT–IS–NOT–TRUE. The unbound sentence is used as an intermediate syntactic structure in order to avoid the duplication of the sentence syntactic rules for relative clauses. Thus both complete sentences and relative clauses use the unbounded sentence S' as a sub-structure, but differ in the treatment of S'_d. In the case of a relative clause, S'_d represents the entities specified by it. In the case of a complete sentence, S'_d is $\mathcal{U} \times \mathcal{U}$ or $\emptyset$ depending on whether the sentence is true or false.[2]

Sentence's hook orientation: rules 4 and 5

(4)	$S^* \leftarrow S'$	$S^*_d = S'_d$
(5)	$S^* \leftarrow S'$	$S^*_d = {S'_d}^{-1}$

The *virtual sentence* (S^*) is an unbound sentence defined with a virtual NP[3] (*gap*) and is therefore a sub-structure of the relative clause. The two potential semantic actions serve to orient the relation (S'_d) depending on whether the virtual NP appears on the subject or the complement side—which determines which side of the relation S'_d holds the entities denoted by the relative clause; see examples (4.30) and (4.31). (Again, the grammar is under-specified for simplicity and does not know which rule to choose.)

Unbound sentences: rules 6 to 8

(6)	$S' \leftarrow$ NP VP	$S'_d = \mathrm{NP}_{\mathrm{after}},\ \mathrm{NP}_{\mathrm{before}} = {\mathrm{VP}_d}^{-1}$
(7)	$S' \leftarrow$ NP NOT VP	$S'_d = \overline{\mathrm{NP}}_{\mathrm{after}},\ \mathrm{NP}_{\mathrm{before}} = {\mathrm{VP}_d}^{-1}$
(8)	$S' \leftarrow$ NP NOT VP	$S'_d = \mathrm{NP}_{\mathrm{after}},\ \mathrm{NP}_{\mathrm{before}} = {\overline{\mathrm{VP}}_d}^{-1}$

An unbound sentence (S') is destined to become a complete sentence (S) or a virtual sentence (S^*). Our grammar covers the basic NP–VP sentence with the optional negation operator that is ambiguous between verb-phrase negation and sentence negation.[4]

'NP VP' The basic sentence structure (rule 6) is handled as in the beginning of this chapter; see example (5.1). The NP maps the verb-phrase, ${\mathrm{VP}_d}^{-1}$, to the denotation of the unbound sentence, S'_d.

'NP NOT VP' The sentence negation (rule 7) is a variation of the previous case, except that S'_d is the complement of the resulting relation.

'NP NOT VP' The verb-phrase negation (rule 8) is a variation of the first case, except that the NP is applied to the complement of VP_d.

[2] Note that the grammar does not keep track of whether S'_d denotes a truth value ($\mathcal{U} \times \mathcal{U}$ or $\emptyset$), a set coerced into a relation ($S \times \mathcal{U}$ or $\mathcal{U} \times S$), or a genuine relation, as this would have increased the complexity of the grammar.

[3] Again, for simplicity this condition is not enforced by the grammar.

[4] Verb negation is left out for simplicity.

5.2.2 Verb phrases

The unique hook of a VP, the relation VP_d, relates each entity qualifying as a subject for the VP to the variable it depends on, or to everything if there is no such variable. (If a VP depends on a variable, VP_d is a genuine relation, not a relation of the type $S \times \mathcal{U}$.)

We account for three types of verb phrases depending on the verb: an ordinary transitive verb, the verb *'to be'*, or the verb *'to have'*.

Ordinary transitive verb: rule 9

(9) VP ← V NP $\qquad \mathrm{VP}_d = \mathrm{NP}_{\mathrm{after}},\ \mathrm{NP}_{\mathrm{before}} = \mathrm{V}_d$

An transitive verb relates every pair of entities (subject–object) that stand in the relationship described by the verb. VP_d is obtained from the mapping of V_d by the NP, as in the example (5.1).

Verb 'to be': rules 10 and 11

(10) VP ← BE NP $\qquad \mathrm{VP}_d = \mathrm{NP}_{\mathrm{after}},\ \mathrm{NP}_{\mathrm{before}} = I$

(11) VP ← BE ADJ $\qquad \mathrm{VP}_d = (\mathrm{ADJ}_d \times \mathcal{U})$

Two derivations produce a VP from the verb *'to be'*.

'BE NP' is handled as 'V VP' with the identity relation (I) as the verb.

'BE ADJ' sets VP_d to the set of entities denoted by ADJ (ADJ_d, the set of entities for which the adjective is true of them) coerced in a relation ($\mathrm{ADJ}_d \times \mathcal{U}$).

Verb 'to have': rules 12 to 14

(12) VP ← HAVE NP $\qquad \mathrm{VP}_d = \mathrm{NP}_{\mathrm{after}},\ \mathrm{NP}_{\mathrm{before}} = \mathtt{possess}$

(13) VP ← HAVE NP $\qquad \mathrm{VP}_d = {\mathrm{NP}_{\mathrm{after}}}^{-1},\ \mathrm{NP}_{\mathrm{before}} = \mathcal{U} \times \mathcal{U}$

(14) VP ← HAVE NP AS R $\qquad \mathrm{VP}_d = \mathrm{NP}_{\mathrm{after}},\ \mathrm{NP}_{\mathrm{before}} = \mathrm{R}_d$

Three derivations produce a VP from the verb *'to have'*. They differ on whether *'have'* denotes an alienable possession, an inalienable possession, or a relation specified.

'HAVE NP' The alienable case is handled as 'V NP' with the possession relation `possess` as verb; see example (4.35) in the previous chapter.

'HAVE NP' The inalienable case counts on the common noun of the NP to be a genuine relation; it is handled as if it meant 'NP OF (subject) be' (here, we mean the existential *'be'* of Descartes). Basically, the NP is applied to the universal relation ($\mathcal{U} \times \mathcal{U}$) and VP_d is set to the inverse of the result, as explained in examples (4.36) to (4.39).

'HAVE NP AS R' is handled as 'V NP' with R in place of the verb; see example (4.40).

5.2.3 Noun phrases

In ERA, an NP is a function that maps a relation to a relation; hence an NP has two hooks, $\mathrm{NP}_{\mathrm{before}}$ and $\mathrm{NP}_{\mathrm{after}}$.

Reflexives: rule 15

(15) $\quad \mathrm{NP} \leftarrow \mathrm{XSELF} \qquad\qquad \mathrm{NP}_{\mathrm{after}} = \Phi(\mathrm{NP}_{\mathrm{before}})$

A reflexive pronoun may appear in the extended object position,[5] not as a subject. (Again, for simplicity the grammar does not enforce this.) A reflexive pronoun aligns itself to the subject, which is done by applying the lowering operator.

Optional binding: rules 16 to 17

(16) $\quad \mathrm{NP} \leftarrow \mathrm{NP}^{(0)} \qquad\qquad \mathrm{NP}^{(0)}_{\mathrm{before}} = \Phi(\mathrm{NP}_{\mathrm{before}})^{-1}$

(17) $\quad \mathrm{NP} \leftarrow \mathrm{NP}^{(0)}$

At this point, the optional binding of the floating variable is done. The relation given in input to the NP, $\mathrm{NP}_{\mathrm{before}}$, may be lowered (or not) before the NP is applied to it. The effect, if the lowering is done, is to bind the free variable to the NP; see example (4.41), and example (4.42) for when the lowering is not done. (For this analogy a relation is viewed as a set depending on a free variable.)

Virtual noun phrase: rule 18

(18) $\quad \mathrm{NP}^{(0)} \leftarrow \perp \qquad\qquad \mathrm{NP}^{(0)}_{\mathrm{after}} = \mathrm{NP}^{(0)}_{\mathrm{before}}$

A virtual NP has simply no effect and maps a relation to itself (see example (4.30)), except that it still has the possibility of binding the floating variable as for the other NP's. Mapping the relation to itself can be viewed as replacing the NP by a free variable, or put in other terms, as simply leaving the variable there instead of quantifying it away.

Quantification: rules 19 to 22

(19) $\quad \mathrm{NP}^{(0)} \leftarrow \mathrm{DET}\ \mathrm{NP}'' \qquad\qquad \mathrm{NP}^{(0)}_{\mathrm{after}} = \mathrm{NP}''_{\mathrm{out}},\ \mathrm{NP}''_{\mathrm{in}} = \mathrm{NP}^{(0)}_{\mathrm{before}} \circ_{\mathrm{DET}} \mathrm{NP}''_{r}$

(20) $\quad \mathrm{NP}^{(0)} \leftarrow \mathrm{NP}'' \qquad\qquad \mathrm{NP}^{(0)}_{\mathrm{after}} = \mathrm{NP}''_{\mathrm{out}},\ \mathrm{NP}''_{\mathrm{in}} = \mathrm{NP}^{(0)}_{\mathrm{before}} \circ_{\mathrm{some}} \mathrm{NP}''_{r}$

(21) $\quad \mathrm{NP}^{(0)} \leftarrow \mathrm{DET}\ \mathrm{OF}\ \mathrm{NP}' \qquad\qquad \mathrm{NP}^{(0)}_{\mathrm{after}} = \mathrm{NP}'_{\mathrm{out}},\ \mathrm{NP}'_{\mathrm{in}} = \mathrm{NP}^{(0)}_{\mathrm{before}} \circ_{\mathrm{DET}} \mathrm{NP}'_{r}$

(22) $\quad \mathrm{NP}^{(0)} \leftarrow \mathrm{NP}' \qquad\qquad \mathrm{NP}^{(0)}_{\mathrm{after}} = \mathrm{NP}'_{\mathrm{out}},\ \mathrm{NP}'_{\mathrm{in}} = \mathrm{NP}^{(0)}_{\mathrm{before}} \circ_{\mathrm{the}} \mathrm{NP}'_{r}$

[5] As complement or complement of the complement, etc.

Each one of these four rules quantifies away the right argument of the input relation $NP^{(0)}_{\text{before}}$ using one of the composition operators to produce the output relation $NP^{(0)}_{\text{after}}$. The rules differ on the type of NP they cover, depending on whether the determiner is specified and if the NP is definite. (NP^* will stand for either NP' or NP'' depending on the rule.)

Three new hooks are introduced: NP^*_r, the restriction relation of the NP; NP^*_{in} and NP^*_{out}, used for the *'of'* prepositional phrases and the possessives. The two latter hooks represent the tricky (or least elegant) part of this semantic interpretation mechanism. They are required for possessives and *'of'* prepositional phrases to take wide scope over the NP because the order in which the syntactic structures are built mismatch the order in which the semantic objects are constructed. Consider the following example.

(5.5) John loves the children of a woman.

1. $xR_1y \leftrightarrow x$ loves the children of y. $\qquad R_1 = \texttt{love} \circ_{\text{the}} \texttt{child-of}$
2. $xR_2y \leftrightarrow x$ loves the children of a woman. $\qquad R_2 = R_1 \circ_{\text{some}} (\texttt{woman} \times \mathcal{U})$
3. John loves the children of a woman. $\qquad \mathcal{U} \times \mathcal{U} = R_2^{-1} \circ_{\text{the1}} (\texttt{john} \times \mathcal{U})$

In this example, the NP *'the children of a woman'* maps the relation `love` to R_2 using two composition operators, and the NP *'a woman'* maps R_1 to R_2. However, when the semantics for *'a woman'* are computed, notice that R_1 (NP_{before} of *'a woman'*), which is needed, is not yet available because the semantics of the NP *'the children of a woman'* (where R_1 is computed) are computed only after the NP *'a woman'* has been handled.

To solve this problem, the two new hooks are used to *memorize* the effect of the internal NP, so the external NP can apply it when ready. For instance, NP_{before} and NP_{after} of *'a woman'* are set to $NP^{(5)}_{\text{in}}$ and $NP^{(5)}_{\text{out}}$ of the external NP in construction (*'the children of a woman'*); then, when the external NP reaches the quantification rules, its $NP^{(0)}_{\text{before}}$ (`love`) is composed to its NP^*_r (`child-of`), the resulting relation is R_1. Now that it is finally available, it is unified with the inherited hook NP^*_{in} (the NP_{before} of *'a woman'*); in effect, *'a woman'* maps R_1 to NP^*_{out}, which becomes $NP^{(0)}_{\text{after}}$ of *'the children of a woman'*. Note that if there is no *'of'* prepositional phrase or possessive to set NP^*_{in} and NP^*_{out}, having them equal each other voids their presence.

Definites: rules 23 to 25

(23) $NP' \leftarrow \text{PRONOUN}$ $\qquad NP'_{\text{out}} = NP'_{\text{in}},\ NP'_r = \mathbf{referent}$

(24) $NP' \leftarrow \text{THE } NP''$ $\qquad NP'_r = \mathbf{referent}(NP''_r)$

(25) $NP' \leftarrow \text{POSS } NP''$ $\qquad NP'_{\text{out}} = \text{POSS}_{\text{out}},\ \text{POSS}_{\text{in}} = NP'_{\text{in}}$

Our fragment distinguishes three types of definites: those introduced by the article *'the'*, pronouns, and possessives. Definites denote a non-empty set—actually, with exactly one element if singular, and more than one if plural—and quantifies universally—unless overridden by a construct 'DET of NP''. The quantification rules are informed of their special status via their syntactic type, NP$'$ instead of NP$''$.

Pronouns The simplest forms of definites are pronouns. Since our fragment doesn't allow pronouns with *'of'* prepositional phrases, like *'those of ...'*, the two hooks $\mathrm{NP}'_{\mathrm{in}}$ and $\mathrm{NP}'_{\mathrm{out}}$ are set to equal each other to void their presence. The restriction hook NP'_r is chosen by an undefined mechanism among a set of available referable expressions; we simply use **referent** to denote the chosen relation. The referent must respect constraints of gender and number (which again are not enforced by the grammar as they should for simplicity). **referent** might be a genuine relation or a set coerced into a relation. (We elaborate on that later when we focus on anaphora in chapter 6.)

The NP's defined with *'the'* may refer back to a set previously mentioned like a pronoun, except that the referent must respect the supplementary constraint of being a sub-relation (subset) of the restriction relation NP''_r. The referent might also simply be the whole relation NP''_r. Unlike the pronoun, the two hooks $\mathrm{NP}'_{\mathrm{in}}$ and $\mathrm{NP}'_{\mathrm{out}}$ are not voided, but inherited from NP$''$, where they were properly defined.

Possessives In our fragment, possessives are handled like *'of'* prepositional phrases,[6] that is, the derivation 'NP 's NP$''$' is handled as 'THE NP$''$ OF NP', and 'POSSADJ NP$''$' as 'THE NP$''$ OF PRONOUN', as illustrated by the two following examples.

(5.6) Every man's child. (The child of every man.)

(5.7) His child. (The child of him.)

So, the semantics for the possessives are built by setting the two hooks $\mathrm{NP}'_{\mathrm{in}}$ and $\mathrm{NP}'_{\mathrm{out}}$ to the corresponding hooks of POSS. (POSS acts like the NP of an *'of'* prepositional phrase, mapping a relation to a relation.)

Integration of 'other': rules 26 and 27

(26) $\mathrm{NP}'' \leftarrow \mathrm{OTHER}\ \mathrm{NP}^{(3)}$ $\qquad \mathrm{NP}''_r = \mathrm{NP}^{(3)}_r \cap \overline{\textbf{referent}}$

(27) $\mathrm{NP}'' \leftarrow \mathrm{NP}^{(3)}$

The word *'other'* is optional. Like a pronoun, 'other' uses **referent**; see discussion of examples (4.45) and (4.46).

[6] Again, this is not a completely satisfying treatment, since there are distinctions between possessives and *'of'* prepositional phrases not captured; for instance, *'the picture of him'* is unambiguous while *'his picture'* might also mean *'the picture that belongs to him'*. In our treatment, the grammar being under-specified, both cases have the two same ambiguous meanings.

Integration of relative clauses: rules 28 and 29

(28) $\mathrm{NP}^{(3)} \leftarrow \mathrm{NP}^{(4)}$ REL $\qquad \mathrm{NP}_r^{(3)} = \mathrm{NP}_r^{(4)} \cap \mathrm{REL}_d$

(29) $\mathrm{NP}^{(3)} \leftarrow \mathrm{NP}^{(4)}$

A relative clause can be optionally integrated into an NP. If so, its semantic interpretation produces a relation, REL_d, constraining the restriction hook (NP_r). REL_d relates each entity verifying the relative clause to the free variable it depends on (or to every element of $\mathcal{U}$ if the relative clause is free of variables). Note that if the relative clause depends on a variable, the NP inherits that variable.

Integration of adjectives: rules 30 and 31

(30) $\mathrm{NP}^{(4)} \leftarrow \mathrm{ADJ}\ \mathrm{NP}^{(4)} \qquad \mathrm{NP}_r^{(4)} \leftarrow \mathrm{NP}_r^{(4)} \cap (\mathrm{ADJ}_d \times \mathcal{U})$

(31) $\mathrm{NP}^{(4)} \leftarrow \mathrm{NP}^{(5)}$

In our fragment, an adjective is viewed as denoting a set of entities,[7] which, when modifying an NP, further constrain the restriction set by intersecting it like the relative clause does. Note that unlike a relative clause, an adjective cannot introduce a dependence to a free variable.

Integration of *'of'* prepositional phrases: rules 32 to 34

(32) $\mathrm{NP}^{(5)} \leftarrow \mathrm{NP}^{(6)}$ OF NP $\qquad \mathrm{NP}_r^{(5)} = \mathrm{NP}_r^{(6)},\ \mathrm{NP}_{\mathrm{out}}^{(5)} = \mathrm{NP}_{\mathrm{after}},\ \mathrm{NP}_{\mathrm{before}} = \mathrm{NP}_{\mathrm{in}}^{(5)}$

(33) $\mathrm{NP}^{(5)} \leftarrow \mathrm{NP}^{(6)}$ OF NP $\qquad \mathrm{NP}_{\mathrm{out}}^{(5)} = \mathrm{NP}_{\mathrm{in}}^{(5)},\ \mathrm{NP}_r^{(5)} = \mathrm{NP}_{\mathrm{after}},\ \mathrm{NP}_{\mathrm{before}} = \mathrm{NP}_r^{(6)}$

(34) $\mathrm{NP}^{(5)} \leftarrow \mathrm{NP}^{(6)} \qquad \mathrm{NP}_{\mathrm{out}}^{(5)} = \mathrm{NP}_{\mathrm{in}}^{(5)}$

In our fragment, an NP can have an optional *'of'* prepositional phrase. The prepositional phrase (or more precisely its NP, the internal NP) can take wide or narrow scope over the NP it modifies (the external NP).

Wide scope To take wide scope, the application of the internal NP is delayed. Its hooks $\mathrm{NP}_{\mathrm{before}}$ and $\mathrm{NP}_{\mathrm{after}}$ are conserved by the external NP (in $\mathrm{NP}_{\mathrm{in}}^{(5)}$ and $\mathrm{NP}_{\mathrm{out}}^{(5)}$ respectively), to be used by the quantification rules.

Narrow scope To take narrow scope, the internal NP maps the current restriction relation of the external NP to its new restriction relation.

The difference between the two options is basically in where the parentheses are put, as illustrated by the following examples, (5.8) for wide scope and (5.9) for narrow scope.

[7] We are aware that this account of adjectives is far from satisfying, but they are not the focus of this thesis.

(5.8) (Every representative) of (a company) came.

$$\mathcal{U} \times \mathcal{U} = [(\mathcal{U} \times \texttt{came}) \circ_{\text{every}} \texttt{representative-of}] \circ_{\text{some1}} (\texttt{company} \times \mathcal{U})$$

(5.9) Every [representative of (a company)] came.

$$\mathcal{U} \times \mathcal{U} = (\mathcal{U} \times \texttt{came}) \circ_{\text{every}} [\texttt{representative-of} \circ_{\text{some1}} (\texttt{company} \times \mathcal{U})]$$

If there is no prepositional phrase, the contribution of the hooks $\text{NP}^{(5)}_{\text{in}}$ and $\text{NP}^{(5)}_{\text{out}}$ is voided by making them equal each other.

Integration of the common noun: rule 35

(35) $\text{NP}^{(6)} \leftarrow \text{CN}$ $\qquad\qquad$ $\text{NP}^{(6)}_r = \text{CN}_d$

The common noun simply specifies the basic restriction relation of the NP, which can be further constrained as seen in the previous rules.

5.2.4 Common nouns

Rules 36 to 40

(36) $\text{CN} \leftarrow \text{N}$ $\qquad\qquad$ $\text{CN}_d = \text{N}_d \times \mathcal{U}$

(37) $\text{CN} \leftarrow \text{N}$ $\qquad\qquad$ $\text{CN}_d = (\text{N}_d \times \mathcal{U}) \cap \texttt{possess}^{-1}$

(38) $\text{CN} \leftarrow \text{R}$ $\qquad\qquad$ $\text{CN}_d = \text{R}_d$

(39) $\text{CN} \leftarrow \text{R}$ $\qquad\qquad$ $\text{CN}_d = (\text{dom}(\text{R}_d) \times \mathcal{U})$

(40) $\text{CN} \leftarrow \text{R}$ $\qquad\qquad$ $\text{CN}_d = (\text{dom}(\text{R}_d) \times \mathcal{U}) \cap \texttt{possess}^{-1}$

A common noun specifies the restriction relation of an NP. The common noun may denote a set of entities (coerced into a relation) or be a genuine relation and denote a set of entities depending on a free variable. For the latter, the free variable is then bound by an *'of'* prepositional phrase, a possessive, or the verb *'to have'*.[8] Accordingly, some common nouns are genuine sets, like *'car'*, and others are genuine relations, like *'child of'* and *'friend of'*. But, it is possible to turn a genuine set like `car` into a genuine relation, `car-of`, by using the relation `possess`; conversely, a genuine relation like `child-of` may be turned into may be turned into a genuine set by applying the domain-projection operator on the relation.[9] In the rules 36 to 37, N denotes a genuine set, while in the rules 38 to 40, R denotes a genuine relation.

[8] The grammar could decide whether the common noun has to or has to not depend on a free variable by looking at the syntactic context, but this would have increased the complexity of the grammar.

[9] Again, we are aware that this treatment is not fully satisfying, but it is not the focus of this thesis.

5.2.5 Relative clauses

Rules 41 to 43

(41) REL $\leftarrow$ PREL S* $\qquad$ $\mathrm{REL}_d = \mathrm{S}^*_d$

(42) REL $\leftarrow$ WHOSE NP$''$ S* $\qquad$ $\mathrm{REL}_d = {\mathrm{NP}''_{\mathrm{out}}}^{-1},\ \mathrm{NP}''_{\mathrm{in}} = \mathrm{S}^*_d \circ_{\mathrm{the}} \mathrm{NP}''_r$

(43) REL $\leftarrow$ WHOSE NP$''$ S* $\qquad$ $\mathrm{REL}_d = \Phi(\mathrm{NP}''_{\mathrm{out}}),\ \mathrm{NP}''_{\mathrm{in}} = \mathrm{S}^*_d \circ_{\mathrm{the}} \mathrm{NP}''_r$

A relative clause constructs a set of entities that possibly depends on a free variable. It does so by having a sentence with a virtual NP which leaves a free variable at its position. The effect is that the sentence's hooks, instead of denoting a truth value ($\mathcal{U} \times \mathcal{U}$ or $\emptyset$), denotes a relation corresponding to the bindings of the free variable which verify the sentence. Note that we assume that the relation is correctly oriented by the ambiguous rules for S*; that is, on the left are the bindings for the virtual NP and on the right, those for the anaphoric NP if any; of course, the two sides are potentially interdependent.

The first rule is for an ordinary relative clause—see example (4.30)—while the last two rules are for relative clauses introduced by *'whose'*—see example (4.31)—which requires two rules to allow an NP defined with *'whose'* the possibility of binding the floating variable, since the binding rules (16) and (17) are not accessible.

5.2.6 Possessives

Rules 44 to 45

(44) POSS $\leftarrow$ NP 's $\qquad$ $\mathrm{POSS}_{\mathrm{out}} = \mathrm{NP}_{\mathrm{after}},\ \mathrm{NP}_{\mathrm{before}} = \mathrm{POSS}_{\mathrm{in}}$

(45) POSS $\leftarrow$ POSSADJ $\qquad$ $\mathrm{POSS}_{\mathrm{out}} = \mathrm{POSS}_{\mathrm{in}} \circ_{\mathrm{the}}$ **referent**

POSS is mainly a syntactic construct denoting an NP in *possessive* position.[10] Therefore, POSS simply conserves the hooks of the NP it is derived from; or if it is a possessive adjective, then it is the same as if the NP was a pronoun, and it is handled as such.

[10] In opposition to *subject* or *object* position.

Chapter 6

Quantification and Anaphora in ERA

In this chapter, we focus on the treatment of quantification and anaphora in ERA.

6.1 Quantification

6.1.1 Scope ambiguities

A problem appears when translating an English sentence into a logical representation: the scope of the NP matters for the meaning but is not clearly specified in English, or say, the rules constraining it are not well understood and apparently rely on context and general knowledge. This problem is analogous to the attempt to compute an arithmetic expression whose parentheses were removed![1] (The analogy is more apparent if the inverse operator is prevented by defining an inverse for each composition operator to allow the addition of an NP from the left.)

(6.1) $$R_1 \circ_D^{-1} R_2 \quad \triangleq \quad ({R_2}^{-1} \circ_D {R_1}^{-1})^{-1}.$$

The sentence below has two potential meanings depending on whether *'a tree'* out-scopes *'every child'* or vice-versa, implying or not that *'every child climbed the same tree'*.

(6.2) Every child climbed a tree.

1. $[(\mathcal{U} \times \texttt{child}) \circ_{\text{every}}^{-1} \texttt{climbed}] \circ_{\text{some1}} (\texttt{tree} \times \mathcal{U}) = \mathcal{U} \times \mathcal{U}$
2. $(\mathcal{U} \times \texttt{child}) \circ_{\text{every}}^{-1} [\texttt{climbed} \circ_{\text{some1}} (\texttt{tree} \times \mathcal{U})] = \mathcal{U} \times \mathcal{U}$

[1] This analogy over-simplifies the issue (or maybe not?) of scope disambiguation since every scope ambiguity cannot be expressed just by playing with the parentheses. A more general approach is to obtain the various meanings by changing the order the NP's quantify a sentence.

In this thesis, we don't address the problem of scope disambiguation. To avoid it, we impose a scope to the NP's according to the syntactic structure of the sentence (The only exception is the *'of'* prepositional phrase taking wide or narrow scope over the NP it modifies.)

- The subject outscopes the complement;
- An NP having a relative clause outscopes every NP inside it;
- An NP having an *'of'* prepositional phrase either outscopes it or is immediately outscoped by it.

6.1.2 Scope structure

In this section, we present an alternative view of our semantics that is helpful for explaining how anaphora works.

The notion of scope captures the order the NP's of a sentence quantify it. Formally, the scope of an operator defines its order of application; that is, the $\circ_D$ operator in the expression $R_1 \circ_D R_2$ outscopes every operator that serves in the construction of R_1 and R_2 (or in a functional representation, $R_1 \circ_D R_2$ outscopes every operator appearing in the subexpressions R_1 and R_2). This notion relates to an NP, since each NP of a sentence may be paired to a $\circ_D$ operator in the semantic representation of the sentence. To accentuate this pairing, an NP is broken in *complement-freed* NP's, where a *complement-freed* NP (NP*) is defined as an NP without a complement or minus its complement. We define the complement of an NP as the object NP of a wide-scope *'of'* prepositional phrase modifying it or the possessor of a possessive modifying it.[2] For example, the NP *'his mother'* is formed out of the two NP*'s *'the mother of'* and *'him'*. The relationship between each composition operator and each NP* is given by the function f denoted by the NP*, which is derived from the composition operator as follows—given that D is the determiner of the NP*, and R, its restriction relation.

$$f(P) \quad = \quad P \circ_D R. \tag{6.3}$$

We define the scope of an NP* as the scope of its $\circ_D$ operator.

Clearly, an NP is expressible as a list of NP*'s, since an NP has at most one complement (that is, in our representation). Moreover, the function an NP denotes is the composition, in order, of the functions its NP*'s denote, starting with its main NP* (actually, the mirror operator might also be applied between two NP*'s of the list if one of the NP*'s binds the floating variable, as explained later).

[2] Note that our representation implicitly allows only one complement.

If we make abstraction of the relative clauses and narrow-scope *'of'* prepositional phrases, the meaning given to a sentence may be expressed as a sequence of relations where each relation is mapped into its successor by an NP* function (for quantifying), the mirror operator (for binding), the inverse operator (for a switch from object to subject position), or the complement operator (for negating), and where the first relation represents the main verb and the last one equals $\emptyset$ or $\mathcal{U} \times \mathcal{U}$ depending on the truth value given to the sentence. The following example illustrates this, using for notation a box to represent a relation and an arrow to represent a function mapping a relation into a relation.

(6.4) Each teacher met one child of every man.

$$\boxed{\texttt{met}} \xrightarrow{\text{one child of}} \boxed{R_1} \xrightarrow{\text{every man}} \boxed{R_2} \xrightarrow{\text{object-subject}} \boxed{R_2{}^{-1}} \xrightarrow{\text{each teacher}} \boxed{\mathcal{U} \times \mathcal{U}}$$

1. $xR_1y \leftrightarrow x$ met one child of y. $\quad R_1 = \texttt{met} \circ_{\text{one}} \texttt{child-of}$
2. $xR_2y \leftrightarrow x$ met one child of every man. $\quad R_2 = R_1 \circ_{\text{every}} (\texttt{man} \times \mathcal{U})$
3. Each teacher met one child of every man. $\quad \mathcal{U} \times \mathcal{U} = R_2{}^{-1} \circ_{\text{every}} (\texttt{teacher} \times \mathcal{U})$

Note that in our box-arrow notation, an NP* out-scopes the NP*'s on its left, since it is applied afterwards. Thus, the order the arrows is determined by the scope given to the NP*'s.

Relative clauses and narrow-scope *'of'* prepositional phrases introduce branches in this otherwise linear representation. In fact, both structures construct a new sequence which final relation serves to define an arrow. A relative clause, like a sentence, produces a sequence of relations which starts with the verb, but ends with a relation holding the entities specified by the relative clause. A narrow-scope *'of'* prepositional phrase modifies the restriction relation of the NP it modifies by producing a sequence that starts with it and ends with its replacement. Here is an example of each.

(6.5) Every man who loves a woman is happy.

$$\boxed{\texttt{love}} \xrightarrow{\text{a woman}} \boxed{R_1}; \quad \boxed{(\mathcal{U} \times \texttt{happy})} \xrightarrow{\text{every } (\text{man} \cap R_1)} \boxed{\mathcal{U} \times \mathcal{U}}$$

1. $xR_1y \leftrightarrow x$ loves a woman. $\quad R_1 = \texttt{love} \circ_{\text{some1}} (\texttt{woman} \times \mathcal{U})$
2. Every man who loves a woman is happy. $\quad \mathcal{U} \times \mathcal{U} = (\mathcal{U} \times \texttt{happy}) \circ_{\text{every}} (\texttt{man} \times \mathcal{U}) \cap R_1$

(6.6) Every parent of a child is happy.

$$\boxed{\texttt{parent-of}} \xrightarrow{\text{a child}} \boxed{R_1}; \quad \boxed{(\mathcal{U} \times \texttt{happy})} \xrightarrow{\text{every } R_1} \boxed{\mathcal{U} \times \mathcal{U}}$$

1. $xR_1y \leftrightarrow x$ parent of a child. $\quad R_1 = \texttt{parent-of} \circ_{\text{some1}} (\texttt{child} \times \mathcal{U})$
2. Every parent of a child is happy. $\quad \mathcal{U} \times \mathcal{U} = (\mathcal{U} \times \texttt{happy}) \circ_{\text{every}} R_1$

6.2 Anaphora

We now focus on the mechanism used for anaphora, for which we use Hirst's definition (Hirst, 1981, p. 4).

> ANAPHORA is the device of making in discourse an ABBREVIATED to some entity (or entities) in the expectation that the perceiver of the discourse will be able to dis-abbreviate the reference and thereby determine the identity of the entity. The reference is called an ANAPHOR, and the entity to which it refers is its REFERENT or ANTECEDENT. A reference and its referent are said to be COREFERENTIAL. The process of determining the referent of an anaphor is called RESOLUTION.

6.2.1 Accessibility

Before going into the types of anaphora, we introduce the notion of accessibility of a set of entities: A is *accessible* from an NP if and only if A is a set depending on a unique free variable[3] to be quantified by the NP; A is *globally accessible* if and only if it is a constant set.

(6.7) Every man who owns a donkey beats it.

(6.8) A man who owns every donkey beats them.

For instance, in the donkey sentence (6.7), *'the donkey owned by each man'* is accessible from *'every man'*; whereas, the set of *'donkeys owned (by a man)'* is globally accessible. In the other donkey sentence (6.8), *'the donkey'* of *'every donkey'* is not accessible from *'a man'*, only the set of *'every donkey'* is.

6.2.2 Classification

Three types of anaphora are distinguished from the *accessibility* of the referent and its scoping relationship with the anaphor. (Note that other constraints apply; for instance, the referent must in general precede the anaphor.)

1. the referent outscopes the anaphor;
2. the referent is accessible globally;
3. the referent is accessible from an NP outscoping the anaphor.

In the first case, the anaphor is simply bound to the variable the referent quantifies. In the second case, the referent is seen as denoting a set used as a definite. The third case is a generalization of the two previous cases.

[3] This constraint of a unique variable is due to our representation.

Here is an example of the two first types of anaphora, which, incidentally, are not exclusive since a referent can both outscope the anaphor and be globally accessible, leading to two equivalent but distinct representations. (The third type of anaphora is exemplified by the donkey sentences.)

(6.9) John_1 loves his_1 wife.

$$\boxed{\texttt{love}} \xrightarrow{\text{the wife of}} \boxed{R_1} \xrightarrow{\text{the } I} \boxed{R_2} \xrightarrow{\text{object-subject}} \boxed{R_2{}^{-1}} \xrightarrow{\text{mirror}} \boxed{\Phi(R_2{}^{-1})^{-1}} \xrightarrow{\text{John}} \boxed{\mathcal{U} \times \mathcal{U}}$$

$$\mathcal{U} \times \mathcal{U} = \Phi([(\texttt{love} \circ_{\texttt{the1}} \texttt{wife-of}) \circ_{\texttt{the1}} I])^{-1} \circ_{\texttt{the1}} (\texttt{john} \times \mathcal{U})$$

1. $xR_1y \leftrightarrow x$ loves the wife of y. $\quad R_1 = \texttt{love} \circ_{\texttt{the1}} \texttt{wife-of}$
2. $xR_2y \leftrightarrow x$ loves the wife of the y. $\quad R_2 = R_1 \circ_{\texttt{the1}} I$
3. John_1 loves his_1 wife. $\quad \mathcal{U} \times \mathcal{U} = \Phi(R_2{}^{-1})^{-1} \circ_{\texttt{the1}} (\texttt{john} \times \mathcal{U})$

$$\boxed{\texttt{love}} \xrightarrow{\text{the wife of}} \boxed{R_1} \xrightarrow{\text{John}} \boxed{R_2} \xrightarrow{\text{object-subject}} \boxed{R_2{}^{-1}} \xrightarrow{\text{John}} \boxed{\mathcal{U} \times \mathcal{U}}$$

$$\mathcal{U} \times \mathcal{U} = \{[\texttt{love} \circ_{\texttt{the1}} \texttt{wife-of}] \circ_{\texttt{the1}} (\texttt{john} \times \mathcal{U})\}^{-1} \circ_{\texttt{the1}} (\texttt{john} \times \mathcal{U})$$

1. $xR_1y \leftrightarrow x$ loves the wife of y. $\quad R_1 = \texttt{love} \circ_{\texttt{the1}} \texttt{wife-of}$
2. $xR_2y \leftrightarrow x$ loves John's wife. $\quad R_2 = R_1 \circ_{\texttt{the1}} (\texttt{john} \times \mathcal{U})$
3. John loves John's wife. $\quad \mathcal{U} \times \mathcal{U} = R_2{}^{-1} \circ_{\texttt{the1}} (\texttt{john} \times \mathcal{U})$

(Note that the $\xrightarrow{\text{the } I}$ arrow is not required by the first alternative, since it denotes the identity function and has no effect ($R_1 = R_2$); yet, it is kept out of considerations to unify the three types of anaphora.)

6.2.3 Resolution

The resolution of the anaphor involves in general four steps, though not all are required by every type of anaphora:

1. find a referent;
2. construct a relation denoting the referent;
3. create a definite denoting the anaphor;
4. apply the mirror operator.

(The construction step is skipped for the first type, since the identity relation is used; the mirror operator application is skipped for the second type, since the object referred is globally accessible, thus does not need to be bound.)

The issue of finding the referent is not addressed by this thesis. The whole next section is devoted to the construction issue. For the rest, the anaphor is handled as a definite over the set captured by the relation representing the referent, and a subsequent binding is done if needed.

(6.10) *John loves a woman*$_1$; her$_1$ name is Mary.

1. $xR_1y \leftrightarrow$ John loves x who is a woman. $\quad R_1 = \texttt{love}^{-1} \circ_{\texttt{the1}} (\texttt{john} \times \mathcal{U}) \cap (\texttt{woman} \times \mathcal{U})$
2. $xR_2y \leftrightarrow$ name of y is Mary. $\quad R_2 = (\mathcal{U} \times \texttt{mary}) \circ_{\texttt{the1}} \texttt{name-of}$
3. her name is Mary. $\quad \mathcal{U} \times \mathcal{U} = R_2 \circ_{\texttt{the1}} R_1$

Given that *'her'* refers to *'a woman'* and that R_1 is the set of *'women loved by John'*, *'her'* is interpreted as a definite having R_1 as restriction.[4]

The issue of applying the mirror operator concerns only the first and third type of anaphora. In their case, the definite denoted by the anaphor depends on a floating variable which has to be bound, which is done by applying the mirror operator.

(6.11) Every man$_1$ loves his$_1$ wife.

$$\boxed{\texttt{love}} \xrightarrow{\text{the wife of}} \boxed{R_1} \xrightarrow{\text{the } I} \boxed{R_2} \xrightarrow{\text{object-subject}} \boxed{{R_2}^{-1}} \xrightarrow{\text{mirror}} \boxed{\Phi({R_2}^{-1})^{-1}} \xrightarrow{\text{every man}} \boxed{\mathcal{U} \times \mathcal{U}}$$

1. $xR_1y \leftrightarrow x$ loves the wife of y. $\quad R_1 = \texttt{love} \circ_{\texttt{the1}} \texttt{wife-of}$
2. $xR_2y \leftrightarrow x$ loves the wife of the y. $\quad R_2 = R_1 \circ_{\texttt{the1}} I$
3. Every man loves his wife. $\quad \mathcal{U} \times \mathcal{U} = \Phi({R_2}^{-1})^{-1} \circ_{\texttt{every}} (\texttt{man} \times \mathcal{U})$

More examples are given in chapter 4. As the example above shows, the binding of the floating variable in itself is direct if the referent immediately outscopes the anaphor. Otherwise the floating variable has to be carried from the anaphor to the NP binding it; this is what we discuss now.

First notice that the function denoting an NP* is actually a mapping from sets to sets (unary relations) extended to a mapping of binary relations into binary relations. This follows from the definition of the $\circ_D$ operator, as illustrated by the following equations.

$$f(P) = P \circ_D R; \tag{6.12}$$

$$[f(P)](x,y) = (D; z; R(z,y); P(x,z)); \tag{6.13}$$

$$[f(P_x)](y) = (D; z; R(z,y); P_x(z)); \tag{6.14}$$

$$[f_{R_y}(P)](x) = (D; z; R_y(z); P(x,z)). \tag{6.15}$$

As equation (6.14) shows, f can be seen as mapping the set P_x into the set $f(P_x)$, both depending on x. (A similar argument may be made about the restriction relation R.) From this perspective, a box-arrow sequence of binary relations is a sequence of sets (unary relations) depending on a variable, where the NP* arrows are set mappings (carrying the dependency forward). However, the mirror operator and the inverse operator operates at the level of binary relations, shaping the sequence with

[4]Note that *'John loves only one woman'* is enforced using R_1 as the restriction set of a singular definite.

respect to the use of the dependency. From the interpretation process, the sequence representing a sentence (or a relative clause) has exactly one object-subject arrow and a subsequent optional mirror arrow, therefore, separating it in three sections: before the object-subject arrow, between the object-subject arrow and the mirror arrow, and after the mirror arrow. The last section might be absent if there is no mirror arrow, and the middle section might be empty if the object-subject and the mirror arrows are consecutive. The dependency (the left side of each relation), is used as follows: in the first section, it carries the subject (this is why there is no point in applying the mirror then[5]); after the object-subject arrow, it carries what remains of the complement and governs the subject, which is, on the right, in quantifying position; after the mirror arrow, it has no use since the dependency is bound. Note that only NP*'s after the object-subject arrow can perform binding; however, any NP* can bound a dependency in its relative clause as its *virtual* subject.

For a sentence, the dependency after the object-subject arrow is left by the NP* preceding the object-subject arrow and has one of the following origin: an anaphor or a dependency inherited from a relative clause. In the case of an anaphor, the NP* preceding the object-subject arrow represents an anaphoric pronoun (or possessive adjective); the function it denotes might depend on a variable (if the referred restriction set depends on a variable), in which case, the dependency is transferred to the right of its output relation (see equation 6.15) but is immediately switched to the left side by the object-subject arrow, where it acts as a free variable. In the other case, the NP preceding the object-subject arrow has a relative clause with a free variable; the function it denotes also depends on a variable, but instead, it comes from the relative clause (equation 6.15).

For a relative clause, the situation is a more complex because of the gap and since a dependency may be introduced from the subject side. The gap (the virtual NP) is handled as a missing arrow in front of the object-subject arrow or at the end, leaving a dependency that reaches the final relation of the sequence on the left or the right. (Note that if the gap is in the object side,[6] the mirror arrow cannot be used as it would prevent the dependency to reach the final relation.) The final relation is then oriented so that the dependency left by the gap takes the left side. The right side may keep (if the gap is on the subject side) a dependency introduced by the object side (as discussed for a sentence) but not bound by the subject side; a dependency might (if the gap is on the object side) also be introduced from the subject side (contrary to a sentence), in which case the dependency originates from the last arrow. In any case, the effect is that the left side of the final relation denotes the elements satisfying the relative clause given any value for its right side.

[5] Technically speaking, there is an exception to that rule in our semantic interpretation process. Reflexive pronouns are bound with the mirror operator before the object-subject reversal, but since the two operations are consecutive, their order may be reversed. (Of course, the side where the mirror operator puts its result switches.)

[6] The NP*'s before the object-subject arrow.

6.2.4 One-variable constraint

The mechanism just described is subject to a *one-variable constraint* inherited from our representation. Basically, the object of any sentence (or relative clause) can depend on at most one variable; the set denoted by a relative clause also can depend on at most one variable. In practice, it is hard to find sentences where such conflicts arise, since the fragment we consider is structurally too simple to generate them. More significantly, an NP* in our fragment can only have one of the following: a possessive, a wide-scope *'of'* prepositional phrase, or a relative clause with an unresolved variable; yet, conflicting sentences don't seem to arise naturally very often. The main limitation is still the tiny coverage of English by our fragment.

6.2.5 Uniqueness effect

If we consider sentences such as (6.16), there is a uniqueness effect implying that *'John loves only one woman'*.

(6.16) John loves a woman; her name is Mary.

This effect is discussed by Kadmon (1987) who links it to the use of anaphora, since the first part of the sentence alone does not imply that *'John loves only one woman'*. In this thesis, this effect is captured by handling an anaphor as a definite, thus using the presuppositions it generates to enforce uniqueness. For instance, a singular NP with *'the'* as article presupposes his restriction set to have a single entity, which for *'her'* implies that referring to the set of *'women loved by John'* presupposes that *'John loves only one woman'*.[7] (Note that since definites require presuppositions to be properly captured, this mechanism falls apart under negation, but we think a proper treatment of definites would remedy to that.)

6.3 Set construction for anaphora

In this section, we finalize our account of anaphora by explaining how the sets referred to by the anaphors are constructed. A popular theory to account for anaphora is DRT (discourse representation theory), introduced independently by Kamp (1981) and Heim (1982). In some sense, they avoid the set construction problem by treating indefinites as free variables instead of quantifiers. We follow a different approach, since mapping their representation into ours would have been tedious if not impossible. In the approach we propose, every NP* is a quantifier associated with a free variable to keep track of its bindings.

[7] To work, this analysis would need to give a special treatment to numerals, since in the sentence *'two men came; they were bald!'*, *'they'* must be understood as *'the two men'* and not simply as *'the men'* in order to enforce that *'exactly two men came'*.

In general, the account for anaphora is straightforward whenever the referent out-scopes the anaphor (just replace the anaphor by the variable bound by the referent), and whenever the referent is a proper noun or a definite accessible globally (just make a copy of the referent). However, if the referent denotes a genuine quantifier that does not out-scope the anaphor, then the anaphor does not refer to the quantifier itself, but rather to the set of entities *involved* in the sentence at the position bound by the quantifier, that is, the maximal set[8] of entities which satisfy the sentence and are in the restriction set of the quantifier. Yet, there are exceptions to that rule, notably when the set turns out to be empty, as for *'no man'*. Then, the set relevant should be the set of *'all men'*. However instead of trying to accommodate for these cases,[9] we assume ideal conditions—no negation and only quantifiers that imply an existential[10]—so that the set thereby constructed is not empty.

Formally, the set S an NP* introduces is determined by its function f and the predicate P it quantifies as follows.

$$f(P) = P \circ_D R; \tag{6.17}$$

$$S(x, y) = \{\, z \mid P(x, z) \wedge R(z, y) \,\}. \tag{6.18}$$

It is easy to verify that the equation (6.18) produces the set needed by the anaphor in the following sentences.

(6.19) A man came; he was bald!

(6.20) Every man who owns a donkey beats it.

That is, for the first sentence, *'a man'* refers to the set of *'men that came'*; and for the second sentence, *'a donkey'* refers to the set of *'donkeys owned by a given man'* which depends on *'a man'* (note that the presuppositions *'only one man came'* and *'every man beats only one donkey'* are derived by treating the anaphor as a singular definite). In general however, the set computed using the equation (6.18) depends on two variables (unless P is constant with respect to x or R is constant with respect to y). This is problematic, since such sets cannot be directly represented in ERA.

Another complication is that the set referred to also depends on the scope relationship between the referent and the anaphor. For instance, an anaphor referring to *'every man'* inside of its scope refers to the variable bound by *'every man'*, while one outside of its scope refers to the set of *'every man'*. Consequently, we must compute the set representing the referent according to the dependencies resolvable from the position of the anaphor. Hence, we need to keep track of the

[8] See Kadmon (1987) for argumentation. Actually, the set might be further restricted by implicit constraints.

[9] This is the major weakness of the method we are proposing. Yet, we don't think this method should be discarded as it is an interesting way to picture the problem, and there are many possible ways it could be fixed.

[10] Meaning that $(D; x; R(x, \ldots); P(x, \ldots)) \Longrightarrow \exists a,\ R(a, \ldots) \wedge P(a, \ldots)$, or mapped into ERA, $R \circ_D P \subseteq R \circ P$.

bindings given to each NP with the dependencies they are subject to. To do so, given a box-arrow sequence of a sentence (or a relative clause), a variable is assigned to each NP* and a system of relations is formed out of the relations of the sequence and the restriction set of each NP*—the relations are related to each other via their arguments which are assigned to the variable of the NP* that binds them. In fact, this system of relations connects together the equation (6.18) of each NP*. Given that there is no negation and every NP* implies an existential, the underlying relation of the system (the conjunction of the relations) holds the bindings given to each NP* and their dependencies (each argument of the relation corresponds to a distinct NP*); hence it will be referred as the bindings relation. If an NP* like *'no man'* were present, its binding set being empty, it would send the whole underlying relation to the empty set, losing the bindings for the other NP*.

The set introduced by an NP* is derived by projecting the bindings relation on its arguments corresponding to the NP* and the dependency that is kept,[11] if any. In the case of a relative clause, the virtual NP must be kept as dependency.

In the following example, we construct the bindings relation for the relative clause of a donkey-like sentence and use it to construct the set introduced by *'one car'* which depends on *'every man'*.

(6.21) ... who likes one car of one of his children drives it.

1.
$$\left\{\begin{array}{rllllllll}
(R_1 = \texttt{like}) & R_1(x,y) & : & x & \text{likes} & y & & & \\
(R_2 = \texttt{car-of}) & R_2(y,z) & : & & & y & \text{car of} & z & \\
(R_3 = R_1 \circ_{\text{one}} R_2) & R_3(x,z) & : & x & \text{likes} & \text{one} & \text{car of} & z & \\
(R_4 = \texttt{child-of}) & R_4(z,x) & : & & & & & z & \text{child of} \;\; x \\
(R_5 = \Phi(R_3 \circ_{\text{one}} R_4)) & R_5(x) & : & x & \text{likes} & \text{one} & \text{car of} & \text{one} & \text{child of} \;\; x
\end{array}\right.$$

2. $R_7(x,y,z) \leftrightarrow R_1(x,y) \wedge R_2(y,z) \wedge R_3(x,z) \wedge R_4(z,x) \wedge R_5(x)$

3. $R_8(y,x) \leftrightarrow \exists z,\ R_7(x,y,z)$ $\qquad R_8 = R_1^{-1} \cap [R_2 \circ (R_3^{-1} \cap R_4)] \cap R_5^{-1}$

(6.22) Every man who likes one car of one of his children drives it.

$$\mathcal{U} \times \mathcal{U} = [(\mathcal{U} \times \texttt{man}) \cap R_5^{-1}] \circ^{-1}_{\text{every}} \Phi(\texttt{drive} \circ_{\text{the1}} R_8)$$

For completeness, we sketch a proof by induction on the NP* that the underlying relation actually holds the bindings—assuming there is no negation and every NP* induces an existential quantifier. Given the box-arrow sequence for a sentence, the bindings for the last NP* are given by the intersection of its restriction relation and the relation it quantifies; both are unary relations. The underlying relation of a system of relations composed of these two relations constraining a variable (the one associated with the NP*) would hold its bindings. Note that since the sentence has no negation and the NP* must imply an existential, the intersection cannot be null. For the induction step, suppose that

[11] To be able to represent the set in a binary relation, only one dependency is kept.

we have a system of relations which hold the bindings of the n right-most NP*'s of the sequence; the bindings for the next NP* (if still any) are obtained as follows: assign an unused variable to it (say z), and add two constraints to the system corresponding to the restriction relation (R) of the NP* and the relation it is applied on (P), with their common argument named z and their other named according to the NP* (among the n previous NP*'s) that binds it (say y and x)—the constraints added are $R(z, y)$ and $P(x, z)$. Observe that this addition permits the expression of the bindings for the NP* depending on the bindings for the other NP*'s. Moreover, it does not affect the underlying relation with respect to the bindings given to the previous NP*'s, since, if z is not considered, the constraint generated by $R(z, y)$ and $P(x, z)$ is subsumed by the constraint $[P \circ_D R](x, y)$ already present, from the previous NP*. It is so, because there is no negation; otherwise the constraint introduced by the previous NP* could be $[P \circ_D R](x, y)$ or its complement. Also, from the fact that every NP* implies an existential, it follows that $P \circ_D R \subseteq P \circ R$; therefore the two new relations have no effect, except on z (only the composition is considered since z is a new variable). Note that the object-subject and the mirror arrows only affect the naming of the variables, allowing us to discard them. The proof is finished. It works as well for relative clauses, except that in this case, the underlying relation captures the bindings of the NP* in dependence to the virtual NP and the unresolved dependencies. Thus in order to obtain the global bindings for the NP* of a relative clause, the binding constraints for the virtual NP and the other non-local dependencies must be added.

Since set construction is not the focus of this thesis, we stop at this point. An obvious enhancement of this technique would be to extend it to negation and arbitrary quantifiers.

A question that arises is whether this model may be used in ERA, since it involves a relation of arbitrary arity, and if so, how to compute the referred set systematically. Here, we show how to do it if there is no long-distance anaphora (see chapter 4). This means that any mirror arrow immediately follows an object-subject arrow and that the set specified by any relative clause is constant. We compute for each NP* its representing set as a function of the floating variable on the right side of the predicate P it quantifies. This divides the NP*'s of a sentence (or relative clause)[12] in two classes: those on the object side for which the representing set is computed in function of the subject and those of the subject side for which the representing set is absolute if in a sentence or depends on the gap if in a relative clause. For each group, to compute the representing set of the most outscoping NP* is direct since R is constant (with respect to its right side) or depends on the same variable as P. (For the object side, if R depends on something it has to be the subject since there is no long-distance anaphora; on the subject side R has to be constant for the same reason.)

$$S(x) \quad = \quad \{\, z \mid \exists y,\ P(x, z) \wedge R(z, x) \,\}. \tag{6.23}$$

[12] Here, we consider NP*'s that are at the same level: in a sentence, those that would participate in its box-arrow representation, that is, the subject, the object, their outscoping noun complements, the outscoping noun complements of their outscoping noun complements, etc...

For the other NP* however, the dependency introduced by its restriction set R must be expressed as a function of the floating variable; C is used for the coercion.

$$S(x) = \{\, z \mid \exists y,\ P(x,z) \wedge R(z,y) \wedge C(y,x)\, \}; \tag{6.24}$$

$$S(x) = \{\, z \mid [P^{-1} \cap (R \circ C)](z,x)\, \}. \tag{6.25}$$

The key observation is that C corresponds to the set introduced by the outscoping NP*, leading to a recursive method of computation. Note that the recursion follows the induction of the proof given above; in fact, the proof could be adapted to show the correctness of this method.

Here are the necessary rules to implement the algorithm just described.

Reflexive pronouns:

(15′) $\mathrm{NP} \leftarrow \mathrm{XSELF}$ $\qquad \mathrm{NP}_{\mathrm{after}} = \Phi(\mathrm{NP}_{\mathrm{before}}),\ \mathrm{NP}_{\mathrm{ref}} = I$

Optional binding of the floating variable by the NP:

(16′) $\mathrm{NP} \leftarrow \mathrm{NP}^{(0)}$ $\qquad \begin{cases} \mathrm{NP}^{(0)}_{\mathrm{before}} = \Phi(\mathrm{NP}_{\mathrm{before}})^{-1} \\ \mathrm{NP}_{\mathrm{ref}} = {\mathrm{NP}^{(0)}_{\mathrm{before}}}^{-1} \cap (\mathrm{NP}^{(0)}_{r} \circ \mathrm{NP}^{(0)}_{\mathrm{prev}}) \end{cases}$

(17′) $\mathrm{NP} \leftarrow \mathrm{NP}^{(0)}$ $\qquad \mathrm{NP}_{\mathrm{ref}} = {\mathrm{NP}^{(0)}_{\mathrm{before}}}^{-1} \cap (\mathrm{NP}^{(0)}_{r} \circ \mathrm{NP}^{(0)}_{\mathrm{prev}})$

Virtual NP of a relative clause:

(18′) $\mathrm{NP}^{(0)} \leftarrow \perp$ $\qquad \mathrm{NP}^{(0)}_{\mathrm{after}} = \mathrm{NP}^{(0)}_{\mathrm{before}},\ \mathrm{NP}^{(0)}_{r} = \mathrm{NP}^{(0)}_{\mathrm{prev}} = I$

Integration of a possessive:

(25′) $\mathrm{NP}' \leftarrow \mathrm{POSS}\ \mathrm{NP}''$ $\qquad \begin{cases} \mathrm{NP}'_{\mathrm{out}} = \mathrm{POSS}_{\mathrm{out}},\ \mathrm{POSS}_{\mathrm{in}} = \mathrm{NP}'_{\mathrm{in}} \\ \mathrm{NP}'_{\mathrm{prev}} = \mathrm{POSS}_{\mathrm{ref}} \end{cases}$

Integration of an optional *'of'* prepositional phrase in the NP:

(32′) $\mathrm{NP}^{(5)} \leftarrow \mathrm{NP}^{(6)}\ \mathrm{OF}\ \mathrm{NP}$ $\qquad \begin{cases} \mathrm{NP}^{(5)}_{r} = \mathrm{NP}^{(6)}_{r},\ \mathrm{NP}^{(5)}_{\mathrm{out}} = \mathrm{NP}_{\mathrm{after}} \\ \mathrm{NP}_{\mathrm{before}} = \mathrm{NP}^{(5)}_{\mathrm{in}},\ \mathrm{NP}^{(5)}_{\mathrm{prev}} = \mathrm{NP}_{\mathrm{ref}} \end{cases}$

(33′) $\mathrm{NP}^{(5)} \leftarrow \mathrm{NP}^{(6)}\ \mathrm{OF}\ \mathrm{NP}$ $\qquad \begin{cases} \mathrm{NP}^{(5)}_{\mathrm{out}} = \mathrm{NP}^{(5)}_{\mathrm{in}},\ \mathrm{NP}^{(5)}_{r} = \mathrm{NP}_{\mathrm{after}} \\ \mathrm{NP}_{\mathrm{before}} = \mathrm{NP}^{(6)}_{r},\ \mathrm{NP}^{(5)}_{\mathrm{prev}} = I \end{cases}$

(34′) $\mathrm{NP}^{(5)} \leftarrow \mathrm{NP}^{(6)}$ $\qquad \mathrm{NP}^{(5)}_{\mathrm{out}} = \mathrm{NP}^{(5)}_{\mathrm{in}},\ \mathrm{NP}^{(5)}_{\mathrm{prev}} = I$

Possessives:

(44′) $\mathrm{POSS} \leftarrow \mathrm{NP}\ \text{'s}$ $\qquad \begin{cases} \mathrm{POSS}_{\mathrm{out}} = \mathrm{NP}_{\mathrm{after}},\ \mathrm{NP}_{\mathrm{before}} = \mathrm{POSS}_{\mathrm{in}} \\ \mathrm{POSS}_{\mathrm{ref}} = \mathrm{NP}_{\mathrm{ref}} \end{cases}$

$$(45') \qquad \text{POSS} \leftarrow \text{POSSADJ} \qquad \begin{cases} \text{POSS}_{\text{out}} = \text{POSS}_{\text{in}} \circ_{\text{the}} \textbf{referent} \\ \text{POSS}_{\text{ref}} = \textbf{referent} \end{cases}$$

Finally, we should mention that using the method of computation here, we believe the problem with negation and *'no man'* could be partly resolved.

Part III

Conclusion

Chapter 7

Abstract Generalized Quantifiers

7.1 Introduction

In this chapter, we show how to extend the BGQ logic of chapter 3 to cover the fragment of English captured by ERA. The interest in doing do is to provide the mathematical tools to study inference and obviousness in the presence of anaphora (which is left for future work). Finally, we look at the expressive power and the computational complexity of ERA by comparing it to other formal languages.

7.2 Extending the BGQ logic

Firstly, we notice that a binary relation is a set depending on a variable. From that perspective, a determiner maps a binary relation to a generalized quantifier *depending on a variable*;[1] we call such a quantifier an *abstract generalized quantifier*. Formally, a determiner D maps the relation R to the abstract generalized quantifier Q ($Q = D(R)$) according to the following equality.

$$\forall d,\ Q(d) \quad = \quad D(\{\, y \mid R(d,y) \,\}). \tag{7.1}$$

Secondly, we extend the notion of superset relation to abstract generalized quantifiers to a notion of conditional superset relation—the condition being the set of pairs of bindings for which the superset relation holds between the two abstract generalized quantifiers.

$$Q_1 \supseteq_{R_3} Q_2 \quad \triangleq \quad R_3 = \{\, (x,y) \mid Q_1(x) \supseteq Q_2(y) \,\}. \tag{7.2}$$

This conditional superset relation reduces to a regular (non-)superset relation if the two abstract

[1] We recall that a determiner, as defined in chapter 2, maps a set into a family of sets.

generalized quantifiers happen to be constant and the condition is not (is) the empty set.

Given $(\forall x,\ Q_1(x) = Q'_1)$ and $(\forall x,\ Q_2(x) = Q'_2)$:

$$Q_1 \supseteq_{\mathcal{U}\times\mathcal{U}} Q_2 \iff Q'_1 \supseteq Q'_2; \tag{7.3}$$

$$Q_1 \supseteq_{\emptyset} Q_2 \iff Q'_1 \not\supseteq Q'_2. \tag{7.4}$$

These equivalences can be rewritten as follows for abstract generalized quantifiers expressible as a determiner mapping a relation that denotes a constant set.

$$D_1(\mathcal{U} \times S_1) \supseteq_{\mathcal{U}\times\mathcal{U}} D_2(\mathcal{U} \times S_2) \iff D_1(S_1) \supseteq D_2(S_2); \tag{7.5}$$

$$D_1(\mathcal{U} \times S_1) \supseteq_{\emptyset} D_2(\mathcal{U} \times S_2) \iff D_1(S_1) \not\supseteq D_2(S_2). \tag{7.6}$$

From this point, we consider only monotone increasing determiners and define the other determiners with respect to them (in order to use the principle for monotone quantifiers).

The conditional subset relation in conjunction with two determiners may serve to define composition operators in ERA.

$$D_1(R_1) \supseteq_{R_3} D_2(R_2) \iff R_3 = f_{D_2}^{D_1}(R_1, R_2). \tag{7.7}$$

In fact, the composition operators of ERA corresponding to monotone determiners are expressible in such a way. Here, D_+ is monotone increasing, and D_- is monotone decreasing.

$$D_+(R_1) \supseteq_{R_3} \mathbf{every}(R_2) \iff R_3 = R_2 \circ_{D_+} (R_1{}^{-1}); \tag{7.8}$$

$$\neg D_-(R_1) \supseteq_{\overline{R_3}} \mathbf{every}(R_2) \iff R_3 = R_2 \circ_{D_-} (R_1{}^{-1}). \tag{7.9}$$

Notice that, since the composition operators of ERA can be modeled, the $\mathcal{RQ}$ function of the BGQ logic is superfluous.

We define the abstract generalized quantifiers logic (AGQ logic) as the boolean algebra of relations in $\mathcal{U}\times\mathcal{U}$ and the algebra of monotone increasing abstract generalized quantifiers (closed under union, intersection, and duality) both linked together by monotone increasing determiners and conditional superset relations—mapping, respectively, relations into abstract generalized quantifiers and vice-versa.

As shown, the AGQ logic can reproduce the composition operators of ERA defined from monotonic determiners. The AGQ logic can also reproduce determiners yielding to quantifiers expressible as the intersection of monotone increasing and decreasing quantifiers, such as *'exactly n'* ($P \circ_{\text{exactly-n}} R = (P \circ_{\text{at-least-n}} R) \cap (P \circ_{\text{at-most-n}} R)$). The inverse operator (see appendix D) and the mirror operator also may be derived in the AGQ logic. Therefore, sentences of ERA made up using these types of composition operators are expressible in the AGQ logic. Consequently, it might be used for the study inference in ERA. (In appendix D, we give some properties of the AGQ logic.)

7.3 Comparison with other formal languages

In this section, we compare ERA (and the AGQ logic) to other logics, namely, the Montagovian syntax (McAllester and Givan, 1992), description logics, and Purdy's $\mathcal{L}_N$ (1991; 1992a; 1992b).

The comparison is done on the quantifying structure of the different logics. Quantification serves to join two relations[2] of arbitrary arity (≥ 1) around one of their arguments (see equation (7.10) written in Moore's notation (1981)). To isolate its role, we assume that the two relations are combined around their first argument, and that they have no other variable in common. To compensate for those restrictions, we assume there is an operator to permute the arguments of a relation, and one for binding one argument of a relation to another one (the equivalent of the mirror operator).

(7.10) $$U(x_1, \ldots, x_k, y_1, \ldots, y_l) = (D; z; R(z, x_1, \ldots, x_k); P(z, y_1, \ldots, y_l))$$

A quantifying structure maps a $(k+1)$-ary relation and an $(l+1)$-ary relation into a $(k+l)$-ary relation. (We call R the domain relation, P the predicate relation, and U the result relation.)

It is possible to classify the logics according to the arity of these three relations in their quantifying structure. First-order logic has no domain relation, or say, R is always the universal predicate. However, typical logics involving generalized quantifiers have no such constraints (Barwise and Cooper, 1981; Does, 1991; Moore, 1981). Yet, some formal languages are characterized by such constraints.

The logic $\mathcal{L}_N$ proposed by Purdy (1991; 1992a; 1992b) has exclusively unary predicates as domain of quantification. His logic is based on the monotonicity of generalized quantifiers as unifying principle, which, he shows, also permits him to account for mass terms (Purdy, 1992b). Also, Purdy is a proponent of surface reasoning, which requires every single inference to be directly expressible in natural language. He shows how to relate $\mathcal{L}_N$ to English. However, anaphora is not covered by his semantic interpretation process. Moreover, we claim, not every sentence involving anaphora can be represented directly (some need to be reformulated, others seem impossible), which breaks the direct the direct correspondence between $\mathcal{L}_N$ and English required for surface reasoning. The problem is that the domain of quantification is restricted to unary predicates. Consequently, sentences like sentence (7.11) are not expressible without involving a reformulation (7.12). (To quantify with *'his children'* directly, the domain of quantification must be a binary predicate.)

[2] Note that in opposition to chapter 4, the relations mentioned here are not necessarily binary.

(7.11) Every man loves his children.

(7.12) Every man (loves-or-is-not-the-father-of) every child.

The problem worsens with *'most'*, as even reformulation seems impossible for sentence (7.13) even though *'most'* is modeled in (Purdy, 1992a).

(7.13) Every man loves most of his children.

In the variable-free Montagovian syntax (or the BGQ logic), R is a unary relation, while P is either a unary relation or a binary relation; the resulting relation U is therefore a truth value or a unary relation.

Some formal languages (Levesque and Brachman, 1987; Borgida and Patel-Schneider, 1994) are specialized in the formal description of classes of objects (concepts). These description logics or terminological logics build their descriptions using concepts (unary predicates) and roles (binary predicates). Typically, the roles specify the domains of quantification and the concepts, the predicate relation. Thus, for their quantifying structures, R is a binary relation and P is a unary relation. Yet, there are exceptions; some of these languages allow role composition, which is a case where both R and P are binary relations.

ERA and the AGQ logic restrict R, P, and U to binary relations. Consequently, ERA encloses the Montagovian syntax and some terminological logics; that is, the objects of these languages can be structurally translated into ERA. A direct implication is that ERA is undecidable (even if a functional representation is adopted and ERA is restricted to first-order definable determiners[3]) as it structurally encompasses a language that Patel-Schneider (1989) has shown undecidable for subsumption (which, in ERA, can be transposed in a satisfiability problem).

Yet, ERA is contained in first-order logic, given that a functional representation is adopted and the language is restricted to first-order determiners. As a proof, it suffices to notice that if R_1 and R_2 are both expressible as a first-order formula with two unbound variables, then so are $R_1 \circ_D R_2$, $\Phi(R_1)$, and ${R_1}^{-1}$. The rest follows by induction.

[3]Determiners that can be defined in first-order logic like *'every'*, *'some'*, *'at least k'*, *'at most k'*, *'exactly k'*. Otherwise, *'most'* defined as *'more than half'* is not first-order definable, as shown by Barwise and Cooper (1981).

Chapter 8

Conclusion

In this thesis, we studied the use of families of sets and binary relations to represent natural language quantification and anaphora.

In part I, we have related the variable-free Montagovian syntax to the theory of generalized quantifiers, by showing that the sentences of the variable-free Montagovian syntax are expressible as subset relationships between families of sets. (Also, we have initiated a mathematical study of subset relationships between generalized quantifiers.) This resulting logic pinpoints the role of transitivity in inference. For future work, it would be interesting to verify the psycholinguistic plausibility of basing human inference on transitivity. The study of the learnability of such a system would also be interesting. We observed that the constraints imposed by the variable-free Montagovian syntax in order to be able to detect unsatisfiability in polynomial time is carried in natural language by the presuppositions, which raises questions about the extent to which they facilitate reasoning. Finally, the BGQ logic may be extended in many ways. While doing this work, we started to investigate a more complete formalism solely based on transitivity; however, the main obstacle is anaphora; this is why we focused on ERA in part II.

In part II, the idea is to represent anaphora using binary relations. Though ERA is restricted to binary relations, it can handle predicates with arbitrary arity given the right model. Instead of seeing a verb as a binary relation, say `love`, it is possible to see it as a set of events, say `love'`, where each event is related to its subject by the relation `subject` and to its object by the relation `object`; in that case, $\texttt{love}' = \texttt{subject}^{-1} \circ [(\texttt{love}' \times \mathcal{U}) \cap I] \circ \texttt{object}$. Using such model we believe it would be possible to handle indirect objects and the other types of complements, even sentential complements like in *'I want to play basketball'*. Yet, to be really interesting for natural language purposes, presuppositions would have to be modeled. (Note that the one-variable constraint is also somehow artificial, since the variable could be used to encode more than one variable, yet this is dependent on the model.) For the study of inference in ERA, we proposed the AGQ logic; it might also be profitable to look at the work on description logics. Finally, the representation of a sentence as a system of relations seems to have imposed itself in the set construction mechanism for anaphora, which raises questions about the direct use of this kind of representation for the semantics of natural language. Naturally, the method should be generalized to handle negative contexts.

Part IV

Appendices

Appendix A

Proofs: Generalized Quantifiers

A.1 Basic properties

The following equivalences and properties follow directly from the definitions and are easy to check (Barwise and Cooper, 1981).

Double negation

(A.1)
$$Q^{\sim} = (\neg Q)\neg = \neg(Q\neg)$$

(A.2)
$$Q = \neg\neg Q = Q\neg\neg = Q^{\sim\sim}.$$

Effect of negation on membership

(A.3)
$$t \in Q \iff \bar{t} \in Q\neg \iff t \notin \neg Q \iff \bar{t} \notin Q^{\sim}.$$

Effect of negation on inclusion

(A.4)
$$Q_1 \subseteq Q_2 \iff Q_1\neg \subseteq Q_2\neg \iff \neg Q_1 \supseteq \neg Q_2 \iff Q_1{}^{\sim} \supseteq Q_2{}^{\sim}.$$

Effect of negation on intersection and union

(A.5)
$$\neg(Q_1 \cap Q_2) = \neg Q_1 \cup \neg Q_2;$$

(A.6)
$$\neg(Q_1 \cup Q_2) = \neg Q_1 \cap \neg Q_2;$$

(A.7)
$$(Q_1 \cap Q_2)\neg = Q_1\neg \cap Q_2\neg;$$

(A.8)
$$(Q_1 \cup Q_2)\neg = Q_1\neg \cup Q_2\neg;$$

(A.9)
$$(Q_1 \cap Q_2)^{\sim} = Q_1{}^{\sim} \cup Q_2{}^{\sim};$$

(A.10)
$$(Q_1 \cup Q_2)^{\sim} = Q_1{}^{\sim} \cap Q_2{}^{\sim}.$$

Effect of negation on some, every and no

$$\neg\mathbf{some}(s) = \mathbf{no}(s); \tag{A.11}$$

$$\mathbf{every}(s)\neg = \mathbf{no}(s); \tag{A.12}$$

$$\mathbf{every}(s)^{\sim} = \mathbf{some}(s); \tag{A.13}$$

$$\neg\mathbf{atleast}_k(s) = \mathbf{atmost}_{k-1}(s). \tag{A.14}$$

A.2 Proof of the principle of monotone quantifiers

Proof of 2.39 Show that given $Q \in \mathcal{Q}_+$, $t \in Q \Longleftrightarrow \mathbf{every}(t) \subseteq Q$.

"$\Longrightarrow$":

$t \in Q \Longrightarrow \forall x \subseteq \mathcal{U},\ t \subseteq x \to x \in Q \Longrightarrow \{\, x \subseteq \mathcal{U} \mid t \subseteq x \,\} \subseteq Q \Longrightarrow \mathbf{every}(t) \subseteq Q$.

"$\Longleftarrow$":

$\mathbf{every}(t) \subseteq Q \Longrightarrow t \in Q$ since $t \in \mathbf{every}(t)$.

Proof of 2.40 Show that given $Q \in \mathcal{Q}_-$, $t \in Q \Longleftrightarrow \mathbf{every}(t) \not\subseteq \neg Q$.

$Q \in \mathcal{Q}_- \Longleftrightarrow \neg Q \in \mathcal{Q}_+$;

therefore, $t \in Q \Longleftrightarrow t \notin \neg Q \Longleftrightarrow \mathbf{every}(t) \not\subseteq \neg Q$.

A.3 Proofs for $\mathcal{D}_S$ and $\mathcal{D}_E$

Proof that $\mathcal{D}_S$ and $\mathcal{D}_E$ are dual Show that $S \in \mathcal{D}_S \Longleftrightarrow S^{\sim} \in \mathcal{D}_E$.

$S \in \mathcal{D}_S \Longleftrightarrow \forall s \subseteq t \subseteq \mathcal{U},\ \forall x \in \mathcal{U},\ [x \in S(s) \leftrightarrow \forall y \in \mathcal{U},\ x \cap s = y \cap s \to y \in S(t)]$;

$\Longleftrightarrow \forall s \subseteq t \subseteq \mathcal{U},\ \forall x \in \mathcal{U},\ [\bar{x} \notin S^{\sim}(s) \leftrightarrow \forall y \in \mathcal{U},\ x \cap s = y \cap s \to \bar{y} \notin S^{\sim}(t)]$;

$\Longleftrightarrow \forall s \subseteq t \subseteq \mathcal{U},\ \forall x' \in \mathcal{U},\ [x' \in S^{\sim}(s) \leftrightarrow \neg\forall y' \in \mathcal{U},\ \bar{x}' \cap s = \bar{y}' \cap s \to y' \notin S^{\sim}(t)]$;

$\Longleftrightarrow \forall s \subseteq t \subseteq \mathcal{U},\ \forall x' \in \mathcal{U},\ [x' \in S^{\sim}(s) \leftrightarrow \exists y' \in \mathcal{U},\ x' \cap s = y' \cap s \wedge y' \in S^{\sim}(t)]$;

$\Longleftrightarrow S^{\sim} \in \mathcal{D}_E$ note that $\bar{x}' \cap s = \bar{y}' \cap s \leftrightarrow x' \cap s = y' \cap s$.

Proof of 2.58 Show that $S \in \mathcal{D}_S, E \in \mathcal{D}_E$ and $s \subseteq t \Longrightarrow S(s) \subseteq S(t)$ and $E(t) \subseteq E(s)$.

$S \in \mathcal{D}_S$ and $s \subseteq t$;

let $x \in S(s) \Longrightarrow \forall y \in \mathcal{U},\ [x \cap s = y \cap s \to y \in S(t)]$;

in particular, choose $y = x \Longrightarrow x \cap s = x \cap s \to x \in S(t) \Longrightarrow x \in S(t)$;

$\Longrightarrow S(s) \subseteq S(t)$ since x is arbitrary ;

moreover, $E \in \mathcal{D}_E \Longrightarrow E^{\sim} \in \mathcal{D}_S \Longrightarrow E^{\sim}(s) \subseteq E^{\sim}(t) \Longrightarrow E(t) \subseteq E(s)$.

Show that $\textbf{some}, \textbf{atleast}_i \in \mathcal{D}_S$ and $\textbf{every} \in \mathcal{D}_E$.

let $t \subseteq \mathcal{U}$, let $s \subseteq t$ and let $x \in \textbf{atleast}_i(s) \Longrightarrow |x \cap s| \geq i$;

$\Longrightarrow \forall y \subseteq \mathcal{U},\ [x \cap s = y \cap s \rightarrow |y \cap t| \geq |y \cap s| = |x \cap s| \geq i]$;

$\Longrightarrow \forall y \in \mathcal{U},\ [x \cap s = y \cap s \rightarrow y \in \textbf{atleast}_i(t)]$;

conversly, $\forall y \in \mathcal{U},\ [x \cap s = y \cap s \rightarrow y \in \textbf{atleast}_i(t)]$;

in particular, choose $y = (x \cap s) \Longrightarrow x \cap s = (x \cap s) \cap s \rightarrow (x \cap s) \in \textbf{atleast}_i(t)$;

$\Longrightarrow (x \cap s) \in \textbf{atleast}_i(t) \Longrightarrow |(x \cap s) \cap t| \geq i$;

$\Longrightarrow |x \cap t| \geq i \Longrightarrow x \in \textbf{atleast}_i(t)$;

therefore, $\textbf{atleast}_i \in \mathcal{D}_S$;

$\Longrightarrow \textbf{some} = \textbf{atleast}_1 \in \mathcal{D}_S$;

$\Longrightarrow \textbf{every} = \textbf{some}^{\sim} \in \mathcal{D}_E$.

A.4 Proof of the intersection lemma

Show that given $S \in \mathcal{D}_S$ and $E \in \mathcal{D}_E$, $S(s) \supseteq E(t) \Longleftrightarrow S(s \cap t) \supseteq E(s \cap t) \Longleftrightarrow S(t) \supseteq E(s)$.

"$\Longrightarrow$":

suppose $S(s) \supseteq E(t)$;

let $x \in E(s \cap t) \Longrightarrow \exists y \subseteq \mathcal{U},\ x \cap s \cap t = y \cap s \cap t \wedge y \in E(t)$;

$\Longrightarrow y \in S(s) \Longrightarrow \forall y' \subseteq \mathcal{U},\ y \cap s \cap t = y' \cap s \cap t \rightarrow y' \in S(s \cap t)$;

in particular, choose $y' = x \Longrightarrow y \cap s \cap t = x \cap s \cap t \rightarrow x \in S(s \cap t)$;

$\Longrightarrow x \in S(s \cap t)$ since $y \cap s \cap t = x \cap s \cap t$;

$\Longrightarrow S(s \cap t) \supseteq E(s \cap t)$.

"$\Longleftarrow$":

suppose $S(s \cap t) \supseteq E(s \cap t)$;

$\Longrightarrow S(s) \supseteq S(s \cap t) \supseteq E(s \cap t) \supseteq E(t)$.

A.5 Proof of the correspondence between $\mathcal{D}_S$ and the symmetric class

Show that $\mathcal{D}_+ \cap \mathcal{D}_S = \mathcal{D}_+ \cap \mathcal{D}_{\text{SYM}}$.

"$\subseteq$":

let $S \in \mathcal{D}_+ \cap \mathcal{D}_S$ and let $s, t \subseteq \mathcal{U}$;

$\Longrightarrow t \in S(s) \leftrightarrow S(s) \supseteq \mathbf{every}(t) \leftrightarrow S(t) \supseteq \mathbf{every}(s) \leftrightarrow s \in S(t)$;

$\Longrightarrow S \in \mathcal{D}_+ \cap \mathcal{D}_{\text{SYM}}$ since s and t are arbitrary .

"$\supseteq$":

let $S \in \mathcal{D}_+ \cap \mathcal{D}_{\text{SYM}}$, let $s \subseteq t \subseteq \mathcal{U}$ and let $x \subseteq \mathcal{U}$;

suppose $x \in S(s) \Longrightarrow x \cap s \in S(s) \Longrightarrow s \in S(x \cap s) \Longrightarrow t \in S(x \cap s) \Longrightarrow x \cap s \in S(t)$;

$\Longrightarrow \forall y \subseteq \mathcal{U},\ x \cap s = y \cap s \rightarrow y \supseteq x \cap s \rightarrow y \in S(t)$;

conversely, suppose $\forall y \subseteq \mathcal{U},\ x \cap s = y \cap s \rightarrow y \in S(t)$;

$\Longrightarrow x \cap s \in S(t) \Longrightarrow t \in S(x \cap s) \Longrightarrow t \cap s \cap x \in S(x \cap s) \Longrightarrow s \in S(x \cap s) \Longrightarrow x \cap s \in S(s)$;

therefore, $S \in \mathcal{D}_+ \cap \mathcal{D}_S$ since s, t and x are arbitrary .

A.6 Proofs for *'every and some'*

Show that $\mathbf{some}(s)^{\sim} = \mathbf{every}(s)$.

$\mathbf{some}(s)^{\sim} = \neg\{\, x \subseteq \mathcal{U} \mid x \cap s \neq \emptyset \,\}\neg = \{\, x \subseteq \mathcal{U} \mid x \cap s = \emptyset \,\}\neg = \{\, x \subseteq \mathcal{U} \mid \overline{x} \cap s = \emptyset \,\}$;

$= \{\, x \subseteq \mathcal{U} \mid s \subseteq x \,\} = \mathbf{every}(s)$.

Show that $\mathbf{some}(s), \mathbf{every}(s) \in \mathcal{Q}_+$.

let $x \in \mathbf{some}(s)$ and let $y \in \mathcal{P}ow(\mathcal{U})$ such that $x \subseteq y$;

$\Longrightarrow x \cap s \neq \emptyset$ and $x \cap s \subseteq y \cap s \Longrightarrow y \cap s \neq \emptyset \Longrightarrow y \in \mathbf{some}(s)$;

therefore, $\mathbf{some}(s) \in \mathcal{Q}_+$;

moreover, $\mathbf{every}(s) = \mathbf{some}(s)^{\sim} \in \mathcal{Q}_+$.

Proof of 2.44 and 2.45 Show that $\mathbf{every}(t) \subseteq \mathbf{every}(s) \Longleftrightarrow \mathbf{some}(s) \subseteq \mathbf{some}(t) \Longleftrightarrow s \subseteq t$.

$\mathbf{every}(t) \subseteq \mathbf{every}(s) \Longleftrightarrow \mathbf{every}(s)^{\sim} \subseteq \mathbf{every}(t)^{\sim}$;

$\Longleftrightarrow \mathbf{some}(s) \subseteq \mathbf{some}(t) \Longleftrightarrow \mathbf{atleast}_1(s) \subseteq \mathbf{atleast}_1(t)$;

$\iff s \subseteq t \vee \mathbf{atleast}_1(s) = \emptyset \iff s \subseteq t \vee s = \emptyset \iff s \subseteq t.$

Proof of 2.46 Show that $\mathbf{every}(s) \subseteq \mathbf{some}(s) \iff |s| \geq 1 \iff s \neq \emptyset$.

"$\Longrightarrow$":

$\mathbf{every}(s) \subseteq \mathbf{some}(s) \iff \{ x \subseteq \mathcal{U} \mid s \subseteq x \} \subseteq \{ x \subseteq \mathcal{U} \mid x \cap s \neq \emptyset \};$

$\Longrightarrow \forall x \subseteq \mathcal{U},\ s \subseteq x \to x \cap s \neq \emptyset \Longrightarrow s \subseteq s \to s \cap s \neq \emptyset;$

$\Longrightarrow s \neq \emptyset \iff |s| \geq 1.$

"$\Longleftarrow$":

$|s| \geq 1 \iff s \neq \emptyset \Longrightarrow \forall x \subseteq \mathcal{U},\ s \subseteq x \to x \cap s = s \neq \emptyset;$

$\Longrightarrow \{ x \subseteq \mathcal{U} \mid s \subseteq x \} \subseteq \{ x \subseteq \mathcal{U} \mid x \cap s \neq \emptyset \} \iff \mathbf{every}(s) \subseteq \mathbf{some}(s).$

Proof of 2.47 Show that $\mathbf{some}(s) \subseteq \mathbf{every}(s) \iff |s| \leq 1$.

"$\Longrightarrow$":

$\mathbf{some}(s) \subseteq \mathbf{every}(s) \iff \{ x \subseteq \mathcal{U} \mid x \cap s \neq \emptyset \} \subseteq \{ x \subseteq \mathcal{U} \mid s \subseteq x \};$

$\Longrightarrow \forall x \subseteq \mathcal{U},\ x \cap s \neq \emptyset \to s \subseteq x \Longrightarrow \forall x \subseteq s,\ x = x \cap s \neq \emptyset \to s \subseteq x \to s = x;$

$\Longrightarrow |s| \leq 1$ since s has no non-trivial subset.

"$\Longleftarrow$":

$|s| \leq 1 \Longrightarrow \forall x \subseteq s,\ x \neq \emptyset \to s = x \Longrightarrow \forall x \subseteq \mathcal{U},\ x \cap s \neq \emptyset \to s = x \cap s \to s \subseteq x;$

$\Longrightarrow \{ x \subseteq \mathcal{U} \mid x \cap s \neq \emptyset \} \subseteq \{ x \subseteq \mathcal{U} \mid s \subseteq x \} \iff \mathbf{some}(s) \subseteq \mathbf{every}(s).$

Proof of 2.48 Show that $\mathbf{every}(s) \subseteq \mathbf{some}(t) \iff |s \cap t| \geq 1$.

$\mathbf{every}(s) \subseteq \mathbf{some}(t) \iff s \in \mathbf{some}(t) \iff |s \cap t| \geq 1.$

Proof of 2.49 Show that $\mathbf{some}(s) \subseteq \mathbf{every}(t) \iff |s \cup t| \leq 1 \vee s = \emptyset \vee t = \emptyset$.

"$\Longrightarrow$":

$\mathbf{some}(s) \subseteq \mathbf{every}(t) \iff \mathbf{atleast}_1(s) \subseteq \mathbf{atleast}_1(t)^{\sim};$

$\Longrightarrow |s \cup t| \leq 1 \vee s = \emptyset \vee t = \emptyset.$

"$\Longleftarrow$":

$|s \cup t| \leq 1 \vee s = \emptyset \vee t = \emptyset;$

$\Longrightarrow \mathbf{atleast}_1(s) \subseteq \mathbf{atleast}_1(t)^{\sim} \vee \mathbf{some}(s) = \emptyset \vee \mathbf{every}(t) = \mathcal{P}ow(\mathcal{U});$

$\Longrightarrow \mathbf{some}(s) \subseteq \mathbf{every}(t).$

A.7 Proofs for *'at least k'*

Definitions

$$\text{(A.15)} \qquad \mathbf{atleast}_n(s) \quad = \quad \{\, x \subseteq \mathcal{U} \mid |s \cap x| \geq n \,\}.$$

Observations

$$\begin{array}{llcl}
\text{(A.16)} & \mathbf{some}(s) & = & \mathbf{atleast}_1(s); \\
\text{(A.17)} & \text{given } |s| < \infty,\ \mathbf{every}(s) & = & \mathbf{atleast}_{|s|}(s); \\
\text{(A.18)} & \mathbf{atleast}_i(s) & \in & \mathcal{Q}_+; \\
\text{(A.19)} & \mathbf{atleast}_i(s) \neq \emptyset & \iff & |s| \geq i; \\
\text{(A.20)} & \mathbf{atleast}_i(s) \supseteq \mathbf{every}(s) & \iff & |s| \geq i; \\
\text{(A.21)} & \mathbf{atleast}_0(s) & = & \mathcal{P}ow(\mathcal{U}); \\
\text{(A.22)} & \emptyset \neq \mathbf{atleast}_i(s) \subseteq \mathbf{atleast}_j(s) \neq \mathcal{P}ow(\mathcal{U}) & \Longrightarrow & i \geq j; \\
\text{(A.23)} & \mathbf{atleast}_i(s) \subseteq \mathbf{atleast}_j(s) & \Longleftarrow & i \geq j; \\
\text{(A.24)} & \mathbf{every}(s) = \mathbf{atleast}_n(s) & \iff & n = |s| < \infty; \\
\text{(A.25)} & \text{given } |s| < \infty,\ \mathbf{atleast}_i(s)^{\sim} & = & \mathbf{atleast}_{|s|-i+1}(s); \\
\text{(A.26)} & \mathbf{atleast}_i(s) \subseteq \mathbf{atleast}_i(s') & \iff & s \subseteq s' \text{ given } \mathbf{atleast}_i(s) \neq \emptyset \wedge i \geq 1.
\end{array}$$

Observations derived from the Intersection Lemma

$$\begin{array}{llcl}
\text{(A.27)} & \mathcal{P}ow(\mathcal{U}) \neq \mathbf{atleast}_i(s) \supseteq \mathbf{atleast}_j(t)^{\sim} \neq \emptyset & \Longrightarrow & |s \cap t| \geq i + j - 1; \\
\text{(A.28)} & \mathbf{atleast}_i(s) \supseteq \mathbf{atleast}_j(t)^{\sim} & \Longleftarrow & |s \cap t| \geq i + j - 1; \\
\text{(A.29)} & \mathcal{P}ow(\mathcal{U}) \neq \mathbf{atleast}_i(s) \supseteq \mathbf{atleast}_j(t) \neq \emptyset & \Longrightarrow & |t \setminus s| \leq j - i; \\
\text{(A.30)} & \mathbf{atleast}_i(s) \supseteq \mathbf{atleast}_j(t) & \Longleftarrow & |t \setminus s| \leq j - i; \\
\text{(A.31)} & \mathcal{P}ow(\mathcal{U}) \neq \mathbf{atleast}_i(s)^{\sim} \supseteq \mathbf{atleast}_j(t) \neq \emptyset & \Longrightarrow & |s \cup t| \leq i + j - 1; \\
\text{(A.32)} & \mathbf{atleast}_i(s)^{\sim} \supseteq \mathbf{atleast}_j(t) & \Longleftarrow & |s \cup t| \leq i + j - 1.
\end{array}$$

Proof of A.16 Show that $\mathbf{some}(s) = \mathbf{atleast}_1(s)$.

$\mathbf{some}(s) = \{\, x \subseteq \mathcal{U} \mid s \cap x \neq \emptyset \,\};$

$\quad = \{\, x \subseteq \mathcal{U} \mid |s \cap x| \geq 1 \,\}$ since $s \cap x \neq \emptyset \iff |s \cap x| \geq 1;$

$= \mathbf{atleast}_1(s)$.

Proof of A.17 Show that given $|s| < \infty$, $\mathbf{every}(s) = \mathbf{atleast}_{|s|}(s)$.

$\forall x \subseteq \mathcal{U},\ s \subseteq x \leftrightarrow s \cap x = s \leftrightarrow |s \cap x| \geq |s|$;

$\Longrightarrow \{\, x \subseteq \mathcal{U} \mid s \subseteq x \,\} = \{\, x \subseteq \mathcal{U} \mid |s \cap x| \geq |s| \,\} \Longleftrightarrow \mathbf{every}(s) = \mathbf{atleast}_{|s|}(s)$.

Proof of A.18 Show that $\mathbf{atleast}_i(s) \in \mathcal{Q}_+$.

let $x \in \mathbf{atleast}_i(s) \Longrightarrow |x \cap s| \geq i$;

$\Longrightarrow \forall y \subseteq \mathcal{U},\ x \subseteq y \rightarrow x \cap s \subseteq y \cap s \rightarrow |y \cap s| \geq |x \cap s| \geq i$;

$\Longrightarrow \forall y \subseteq \mathcal{U},\ x \subseteq y \rightarrow y \in \mathbf{atleast}_i(s)$;

$\Longrightarrow \mathbf{atleast}_i(s) \in \mathcal{Q}_+$ since x arbitrary .

Proof of A.19 Show that $\mathbf{atleast}_i(s) \neq \emptyset \Longleftrightarrow |s| \geq i$.

"$\Longrightarrow$":

$\mathbf{atleast}_i(s) \neq \emptyset \Longrightarrow \mathcal{U} \in \mathbf{atleast}_i(s)$ since $\emptyset \neq \mathbf{atleast}_i(s) \in \mathcal{Q}_+$;

$\Longrightarrow |s| = |\mathcal{U} \cap s| \geq i$.

"$\Longleftarrow$":

$|s| \geq i \Longrightarrow s \in \mathbf{atleast}_i(s)$ since $|s \cap s| = |s| \geq i \Longrightarrow \mathbf{atleast}_i(s) \neq \emptyset$.

Proof of A.20 Show that $\mathbf{atleast}_i(s) \supseteq \mathbf{every}(s) \Longleftrightarrow |s| \geq i$.

$\mathbf{atleast}_i(s) \supseteq \mathbf{every}(s) \Longleftrightarrow s \in \mathbf{atleast}_i(s) \Longleftrightarrow |s| \geq i$.

Proof of A.21 Show that $\mathbf{atleast}_0(s) = \mathcal{P}ow(\mathcal{U})$.

$\mathbf{atleast}_0(s) = \{\, x \in \mathcal{U} \mid |x \cap s| \geq 0 \,\} = \{\, x \in \mathcal{U} \mid \mathbf{true} \,\} = \mathcal{P}ow(\mathcal{U})$.

Proof of A.22 and A.23 Show that given $\mathbf{atleast}_i(s) \neq \emptyset$, $\mathbf{atleast}_i(s) \subseteq \mathbf{atleast}_j(s) \Longleftrightarrow i \geq j$.

"$\Longrightarrow$":

$\emptyset \neq \mathbf{atleast}_i(s) \subseteq \mathbf{atleast}_j(s)$;

$\Longrightarrow |s| \geq i$ since $\mathbf{atleast}_i(s) \neq \emptyset \Longrightarrow \exists s' \subseteq s,\ |s'| = i$;

$\Longrightarrow s' \in \mathbf{atleast}_i(s)$ since $|s' \cap s| = |s| = i \geq i$;

$\Longrightarrow s' \in \mathbf{atleast}_j(s)$ since $\mathbf{atleast}_i(s) \subseteq \mathbf{atleast}_j(s)$;

$\Longrightarrow i = |s'| = |s' \cap s| \geq j$.

"$\Longleftarrow$":

$i \geq j \Longrightarrow \forall n,\ n \geq i \rightarrow n \geq j \Longrightarrow \forall x \subseteq \mathcal{U},\ |s \cap x| \geq i \rightarrow |s \cap x| \geq j$;

$\Longrightarrow \{\, x \subseteq \mathcal{U} \mid |s \cap x| \geq i \,\} \subseteq \{\, x \subseteq \mathcal{U} \mid |s \cap x| \geq j \,\} \Longleftrightarrow \mathbf{atleast}_i(s) \subseteq \mathbf{atleast}_j(s)$.

Proof of A.24 Show that given $|s| < \infty$, $\mathbf{every}(s) = \mathbf{atleast}_n(s) \Longleftrightarrow n = |s|$.

"$\Longrightarrow$":

$\mathbf{every}(s) = \mathbf{atleast}_n(s) \Longrightarrow \emptyset \neq \mathbf{every}(s) = \mathbf{atleast}_{|s|}(s) = \mathbf{atleast}_n(s)$;

$\Longrightarrow \emptyset \neq \mathbf{atleast}_{|s|}(s) \subseteq \mathbf{atleast}_n(s)$ and $\emptyset \neq \mathbf{atleast}_n(s) \subseteq \mathbf{atleast}_{|s|}(s)$;

$\Longrightarrow |s| \geq n$ and $n \geq |s| \Longrightarrow |s| = n$.

"$\Longleftarrow$":

$n = |s| \Longrightarrow \mathbf{atleast}_n(s) = \mathbf{atleast}_{|s|}(s) = \mathbf{every}(s)$.

Proof of A.25 Show that given $|s| < \infty$, $\mathbf{atleast}_i(s)^{\sim} = \mathbf{atleast}_{|s|-i+1}(s)$.

$\mathbf{atleast}_i(s)^{\sim} = \{\, x \subseteq \mathcal{U} \mid |s \cap \overline{x}| \not\geq i \,\}$;

$= \{\, x \subseteq \mathcal{U} \mid |s \cap x| + |s \cap \overline{x}| < i + |s \cap x| \,\}$;

$= \{\, x \subseteq \mathcal{U} \mid |s| < i + |s \cap x| \,\} = \{\, x \subseteq \mathcal{U} \mid |s \cap x| \geq |s| - i + 1 \,\}$;

$= \mathbf{atleast}_{|s|-i+1}(s)$.

Proof of A.26 Show that given $\mathbf{atleast}_i(s) \neq \emptyset$ and $i \geq 1$,

$\mathbf{atleast}_i(s) \subseteq \mathbf{atleast}_i(s') \Longleftrightarrow s \subseteq s'$.

"$\Longrightarrow$":

$\mathbf{atleast}_i(s) \neq \emptyset \Longrightarrow |s| \geq i \geq 1 \Longrightarrow s \neq \emptyset$;

let $\Omega_s^i = \{\, x \subseteq s \mid |x| = i \,\} \Longrightarrow \Omega_s^i \neq \emptyset$ since $|s| \geq i \geq 1$;

also, note that $(\cup_{x \in \Omega_s^i} x) = s$;

moreover, $\Omega_s^i \subseteq \mathbf{atleast}_i(s)$ since $x \in \Omega_s^i \Longrightarrow |x \cap s| = |x| = i \geq i$;

$\Longrightarrow \Omega_s^i \subseteq \mathbf{atleast}_i(s')$;

therefore, $x \in \Omega_s^i \subseteq \mathbf{atleast}_i(s') \rightarrow |x \cap s'| \geq i = |x| \rightarrow x \subseteq s'$;

$\Longrightarrow s = (\cup_{x \in \Omega_s^i} x) \subseteq s' \Longrightarrow s \subseteq s'$.

"$\Longleftarrow$":

$s \subseteq s' \Longrightarrow x \cap s \subseteq x \cap s' \Longrightarrow |x \cap s| \leq |x \cap s'| \Longrightarrow |x \cap s| \geq i \rightarrow |x \cap s'| \geq i$;

$\Longrightarrow \{\, x \subseteq \mathcal{U} \mid |x \cap s| \geq i \,\} \subseteq \{\, x \subseteq \mathcal{U} \mid |x \cap s'| \geq i \,\} \Longleftrightarrow \mathbf{atleast}_i(s) \subseteq \mathbf{atleast}_i(s')$.

Proof of A.27 and A.28 Show that given $\mathcal{P}ow(\mathcal{U}) \neq \mathbf{atleast}_i(s)$ and $\mathbf{atleast}_j(t)^{\sim} \neq \emptyset$,

$\mathbf{atleast}_i(s) \supseteq \mathbf{atleast}_j(t)^{\sim} \Longleftrightarrow |s \cap t| \geq i + j - 1$.

$\mathbf{atleast}_i(s) \supseteq \mathbf{atleast}_j(t)^{\sim}$;

$\iff \mathbf{atleast}_i(s \cap t) \supseteq \mathbf{atleast}_j(s \cap t)^{\sim}$ using the Intersection Lemma;

$\iff \mathbf{atleast}_i(s \cap t) \supseteq \mathbf{atleast}_{|s \cap t|-j+1}(s \cap t)$;

$\iff i \leq |s \cap t| - j + 1$;

since $\mathcal{P}ow(\mathcal{U}) \neq \mathbf{atleast}_i(s) \supseteq \mathbf{atleast}_i(s \cap t)$ and $\mathbf{atleast}_j(s \cap t)^{\sim} \supseteq \mathbf{atleast}_j(t)^{\sim} \neq 0$;

$\iff |s \cap t| \geq i + j - 1$.

Proof of A.29 and A.30 Show that given $\mathcal{P}ow(\mathcal{U}) \neq \mathbf{atleast}_i(s)$ and $\mathbf{atleast}_j(t) \neq \emptyset$,

$\mathbf{atleast}_i(s) \supseteq \mathbf{atleast}_j(t) \iff |t \setminus s| \leq j - i$.

$\mathbf{atleast}_i(s) \supseteq \mathbf{atleast}_j(t) = \mathbf{atleast}_{|t|-j+1}(t)^{\sim}$;

$\iff |s \cap t| \geq i + (|t| - j + 1) - 1$ using the previous result;

$\iff |t \setminus s| \leq j - i$.

Proof of A.31 and A.32 Show that given $\mathcal{P}ow(\mathcal{U}) \neq \mathbf{atleast}_i(s)^{\sim}$ and $\mathbf{atleast}_j(t) \neq \emptyset$,

$\mathbf{atleast}_i(s)^{\sim} \supseteq \mathbf{atleast}_j(t) \iff |s \cup t| \leq i + j - 1$.

$\mathbf{atleast}_{|s|-i+1}(s) = \mathbf{atleast}_i(s)^{\sim} \supseteq \mathbf{atleast}_j(t) = \mathbf{atleast}_{|t|-j+1}(t)^{\sim}$;

$\iff |s \cap t| \geq (|s| - i + 1) + (|t| - j + 1) - 1$;

$\iff |s \cup t| \leq i + j - 1$.

A.8 Properties of $\Pi_d(R)$

The following properties are immediate from the definition of Π_d:

(A.33) $$\Pi_d(\emptyset) = \emptyset;$$

(A.34) $$\Pi_d(\bar{R}) = \overline{\Pi_d(R)};$$

(A.35) $$\Pi_d(R_1 \cap R_2) = \Pi_d(R_1) \cap \Pi_d(R_2).$$

This implies that Π_d is a ring homomorphism from the boolean algebra $\mathcal{B}ool(\mathcal{P}ow(\mathcal{U} \times \mathcal{U}))$ to $\mathcal{B}ool(\mathcal{P}ow(\mathcal{U}))$. Moreover, from the previous properties, it follows that:

(A.36) $$\Pi_d(\mathcal{U} \times \mathcal{U}) = \mathcal{U};$$

(A.37) $$\Pi_d(R_1 \cup R_2) = \Pi_d(R_1) \cup \Pi_d(R_2);$$

(A.38) $$R_1 \subseteq R_2 \implies \Pi_d(R_1) \subseteq \Pi_d(R_2).$$

Note that this last property of Π_d makes it also a functor from $\mathcal{C}ategory(\mathcal{P}ow(\mathcal{U} \times \mathcal{U}), \subseteq)$ to $\mathcal{C}ategory(\mathcal{P}ow(\mathcal{U}), \subseteq)$.

Show that $R_1 \subseteq R_2 \Longrightarrow \Pi_d(R_1) \subseteq \Pi_d(R_2)$.

$R_1 \subseteq R_2 \Longleftrightarrow R_1 \cap \bar{R}_2 = \emptyset$;

$\Longrightarrow \Pi_d(R_1 \cap \bar{R}_2) = \emptyset \Longrightarrow \Pi_d(R_1) \cap \overline{\Pi_d(R_2)} = \emptyset \Longrightarrow \Pi_d(R_1) \subseteq \Pi_d(R_2)$.

Another property of $\Pi_d(R)$ that is not related to the others, but also follows directly from the definition, is that:

(A.39) $$d' \in \Pi_d(R) \iff d \in \Pi_{d'}(R^{-1}).$$

A.9 Properties of $\mathcal{RQ}_R(Q)$

Properties of $\mathcal{RQ}_R$

Similarly as for Π_d, the following properties derive from the definition of $\mathcal{RQ}_R$:

(A.40) $$\mathcal{RQ}_R(\emptyset) = \emptyset;$$

(A.41) $$\mathcal{RQ}_R(\neg Q) = \overline{\mathcal{RQ}_R(Q)};$$

(A.42) $$\mathcal{RQ}_R(Q_1 \cap Q_2) = \mathcal{RQ}_R(Q_1) \cap \mathcal{RQ}_R(Q_2).$$

So, this implies that $\mathcal{RQ}_R$ is a ring homomorphism from $\mathcal{B}ool(\mathcal{P}ow(\mathcal{P}ow(\mathcal{U})))$ to $\mathcal{B}ool(\mathcal{P}ow(\mathcal{U}))$. It is also a functor from $\mathcal{C}ategory(\mathcal{P}ow(\mathcal{P}ow(\mathcal{U})), \subseteq)$ to $\mathcal{C}ategory(\mathcal{P}ow(\mathcal{U}), \subseteq)$ and the following properties are derived from the previous ones.

(A.43) $$\mathcal{RQ}_R(\mathcal{P}ow(\mathcal{U})) = \mathcal{U};$$

(A.44) $$\mathcal{RQ}_R(Q_1 \cup Q_2) = \mathcal{RQ}_R(Q_1) \cup \mathcal{RQ}_R(Q_2);$$

(A.45) $$Q_1 \subseteq Q_2 \Longrightarrow \mathcal{RQ}_R(Q_1) \subseteq \mathcal{RQ}_R(Q_2).$$

Properties of $\mathcal{RQ}_?(Q)$

$\mathcal{RQ}_R(Q)$ can also be seen as a function of R by fixing Q. Note that if Q is generated by a single object ($Q = \mathbf{every}(\{d\}) = \mathbf{some}(\{d\})$ with $d \in \mathcal{U}$) then $\mathcal{RQ}_R(Q) = \Pi_d(R^{-1})$. Therefore, in this case, $\mathcal{RQ}_R(Q)$ as a function of R is a ring homomorphism and a functor as for Π_d.

(A.46) $$\mathcal{RQ}_R(\mathbf{every}(\{d\})) = \Pi_d(R^{-1}).$$

In the general case however, $\mathcal{RQ}_R(Q)$ as a function of R is not a ring homomorphism. But it is still a functor from $\mathcal{C}ategory(\mathcal{P}ow(\mathcal{U}\times\mathcal{U}),\ \subseteq)$ to $\mathcal{C}ategory(\mathcal{P}ow(\mathcal{U}),\ \subseteq)$ if Q is non-trivial monotone increasing and a functor from $\mathcal{C}ategory(\mathcal{P}ow(\mathcal{U}\times\mathcal{U}),\ \subseteq)$ to $\mathcal{C}ategory(\mathcal{P}ow(\mathcal{U}),\ \supseteq)$ if Q is non-trivial monotone decreasing. ($\mathcal{Q}^*_+$ (and $\mathcal{Q}^*_-$) are the classes of non-trivial monotone increasing (and decreasing) quantifiers.)

(A.47) $$Q \in \mathcal{Q}^*_+ \implies \mathcal{RQ}_\emptyset(Q) = \emptyset \text{ and } \mathcal{RQ}_{\mathcal{U}\times\mathcal{U}}(Q) = \mathcal{U};$$

(A.48) $$Q \in \mathcal{Q}_+ \text{ and } R_1 \subseteq R_2 \implies \mathcal{RQ}_{R_1}(Q) \subseteq \mathcal{RQ}_{R_2}(Q);$$

(A.49) $$Q \in \mathcal{Q}^*_- \implies \mathcal{RQ}_\emptyset(Q) = \mathcal{U} \text{ and } \mathcal{RQ}_{\mathcal{U}\times\mathcal{U}}(Q) = \emptyset;$$

(A.50) $$Q \in \mathcal{Q}_- \text{ and } R_1 \subseteq R_2 \implies \mathcal{RQ}_{R_1}(Q) \supseteq \mathcal{RQ}_{R_2}(Q).$$

Properties of $\mathcal{RQ}_R(Q)$ vs Negation

(A.51) $$\mathcal{RQ}_R(Q) = \overline{\mathcal{RQ}_R(\neg Q)} = \mathcal{RQ}_{\bar{R}}(Q\neg) = \overline{\mathcal{RQ}_{\bar{R}}(Q^{\sim})}.$$

Properties of $\mathcal{RQ}_R(Q)$ on the boundaries

(A.52) $$\mathcal{RQ}_R(\emptyset) = \emptyset;$$

(A.53) $$\mathcal{RQ}_R(\mathcal{P}ow(\mathcal{U})) = \mathcal{U};$$

(A.54) $$\mathcal{RQ}_\emptyset(Q) = \emptyset \text{ if } \emptyset \notin Q,\ \mathcal{U} \text{ otherwise };$$

(A.55) $$\mathcal{RQ}_{\mathcal{U}\times\mathcal{U}}(Q) = \mathcal{U} \text{ if } \mathcal{U} \in Q,\ \emptyset \text{ otherwise }.$$

Properties of $\mathcal{RQ}_R(\textbf{some}(s))$ and $\mathcal{RQ}_R(\textbf{every}(s))$

(A.56) $$\mathcal{RQ}_R(Q) \in \textbf{some}(s) \implies \mathcal{RQ}_{R^{-1}}(\textbf{some}(s)) \in Q;$$

(A.57) $$\mathcal{RQ}_R(\textbf{every}(s)) \in Q \implies \mathcal{RQ}_{R^{-1}}(Q) \in \textbf{every}(s);$$

(A.58) $$\mathcal{RQ}_{R_1\cup R_2}(\textbf{some}(s)) = \mathcal{RQ}_{R_1}(\textbf{some}(s)) \cup \mathcal{RQ}_{R_2}(\textbf{some}(s));$$

(A.59) $$\mathcal{RQ}_{R_1\cap R_2}(\textbf{every}(s)) = \mathcal{RQ}_{R_1}(\textbf{every}(s)) \cap \mathcal{RQ}_{R_2}(\textbf{every}(s)).$$

The two first properties capture the scope movement of the existential quantifier, inward, and of the universal quantifier, outward. The last two properties follow from the fact that the existential quantifier distributes over the disjunction and the universal one over the conjunction.

Appendix B

Proofs: variable-free Montagovian syntax and BGQ logic

B.1 Bijection between the formulas of the BGQ logic and the Montagovian syntax

Here, we show there is a bijection that translates formulas from one formalism to the other while preserving their semantics. To do so, a bijection is defined recursively on the basic objects of the two languages.

$$(R\ (\text{some } s)) \xleftrightarrow{\text{corresponds}} \langle R\ [\text{some } s]\rangle; \tag{B.1}$$

$$(R\ (\text{every } s)) \xleftrightarrow{\text{corresponds}} \langle R\ [\text{every } s]\rangle; \tag{B.2}$$

$$(\text{every } s\ t) \xleftrightarrow{\text{corresponds}} [\text{every } s] \sqsupseteq [\text{every } t]; \tag{B.3}$$

$$(\text{some } s\ t) \xleftrightarrow{\text{corresponds}} [\text{some } s] \sqsupseteq [\text{every } t]; \tag{B.4}$$

$$\exists s \xleftrightarrow{\text{corresponds}} [\text{some } s] \sqsupseteq [\text{every } s]; \tag{B.5}$$

$$(\text{at-most-one } s) \xleftrightarrow{\text{corresponds}} [\text{every } s] \sqsupseteq [\text{some } s]. \tag{B.6}$$

Note here that we don't technically have a bijection, since the formulas of the type [every s] $\sqsupseteq$ [some t] have no equivalent in the variable-free fragment of the Montagovian syntax. Yet, this is not a problem, as the completeness result requires the class expressions to be determined; so the problematic formula is trivially true if $s = \emptyset$ or $t = \emptyset$ or means that $s = t$ and $|s| = 1$, which is expressible in the variable-free fragment of the Montagovian syntax (this follows since $\mathbf{every}(s) \supseteq \mathbf{some}(t) \iff |s \cup t| \leq 1 \vee s = \emptyset \vee t = \emptyset$). The correspondence given preserves the semantics of the formula. Again, this is shown recursively by checking that the corresponding objects have the same semantics.

$\varphi((R\ (\text{every } s)))$;

$$= \{\, d \in \mathcal{U} \mid \forall d' \in \varphi(s),\ (d, d') \in \varphi(R)\,\} = \{\, d \in \mathcal{U} \mid \forall d' \in \varphi(s),\ d' \in \Pi_d(\varphi(R))\,\};$$

$$= \{\, d \in \mathcal{U} \mid \varphi(s) \subseteq \Pi_d(\varphi(R))\,\} = \{\, d \in \mathcal{U} \mid \Pi_d(\varphi(R)) \in \mathbf{every}(\varphi(s))\,\};$$

$$= \mathcal{RQ}_{\varphi(R)}(\mathbf{every}(\varphi(s))) = \mathcal{RQ}_{\varphi(R)}(\varphi([\text{every } s])) = \varphi(\langle R\ [\text{every } s]\rangle).$$

$\varphi((R\ (\text{some } s)))$;

$$= \{\, d \in \mathcal{U} \mid \exists_{d' \in \varphi(s)} (d, d') \in \varphi(R)\,\} = \{\, d \in \mathcal{U} \mid \exists_{d' \in \varphi(s)} d' \in \Pi_d(\varphi(R))\,\};$$

$$= \{\, d \in \mathcal{U} \mid \varphi(s) \cap \Pi_d(\varphi(R)) \neq \emptyset\,\} = \{\, d \in \mathcal{U} \mid \Pi_d(\varphi(R)) \in \mathbf{some}(\varphi(s))\,\};$$

$$= \mathcal{RQ}_{\varphi(R)}(\mathbf{some}(\varphi(s))) = \mathcal{RQ}_{\varphi(R)}(\varphi([\text{some } s])) = \varphi(\langle R\ [\text{some } s]\rangle).$$

$\varphi((\text{every } s\ t)) = \mathbf{true}$;

$$\iff \varphi(s) \subseteq \varphi(t) \iff \varphi(t) \in \mathbf{every}(\varphi(s)) \iff \mathbf{every}(\varphi(t)) \subseteq \mathbf{every}(\varphi(s));$$

$$\iff \varphi([\text{every } t]) \subseteq \varphi([\text{every } s]) \iff \varphi([\text{every } s] \sqsupseteq [\text{every } t]) = \mathbf{true}.$$

$\varphi((\text{some } s\ t)) = \mathbf{true}$;

$$\iff \varphi(s) \cap \varphi(t) \neq \emptyset \iff \varphi(t) \in \mathbf{some}(\varphi(s)) \iff \mathbf{every}(\varphi(t)) \subseteq \mathbf{some}(\varphi(s));$$

$$\iff \varphi([\text{every } t]) \subseteq \varphi([\text{some } s]) \iff \varphi([\text{some } s] \sqsupseteq [\text{every } t]) = \mathbf{true}.$$

$\varphi(\exists s) = \mathbf{true}$;

$$\iff \varphi(s) \neq \emptyset \iff \mathbf{every}(\varphi(s)) \subseteq \mathbf{some}(\varphi(s));$$

$$\iff \varphi([\text{every } s]) \subseteq \varphi([\text{some } s]) \iff \varphi([\text{some } s] \sqsupseteq [\text{every } s]) = \mathbf{true}.$$

$\varphi((\text{at-most-one } s)) = \mathbf{true}$;

$$\iff |\varphi(s)| \leq 1 \iff \mathbf{some}(\varphi(s)) \subseteq \mathbf{every}(\varphi(s));$$

$$\iff \varphi([\text{some } s]) \subseteq \varphi([\text{every } s]) \iff \varphi([\text{every } s] \sqsupseteq [\text{some } s]) = \mathbf{true}.$$

B.2 Soundness of the BGQ logic inferences rules

Now to show that the inference rules are equivalent under local inferences, we need to prove that the inference rules for the BGQ logic are indeed sound. This is done using the general properties of the generalized quantifiers shown in chapter 2.

Show that "$\vdash [\text{some } c] \sqsupseteq [\text{every } c] \sqsupseteq [\text{some } c]$" is sound.

$|\varphi(c)| = 1 \Longrightarrow \textbf{some}(\varphi(c)) = \textbf{every}(\varphi(c));$

$\Longrightarrow \textbf{some}(\varphi(c)) \supseteq \textbf{every}(\varphi(c)) \supseteq \textbf{some}(\varphi(c));$

$\Longrightarrow \varphi([\text{some } s] \sqsupseteq [\text{every } c] \sqsubseteq [\text{some } c]) = \textbf{true}.$

Show that "$\vdash Q \sqsupseteq Q$" is sound.

$\varphi(Q) = \varphi(Q) \Longrightarrow \varphi(Q) \supseteq \varphi(Q) \Longrightarrow \varphi(Q \sqsupseteq Q).$

Show that "$Q_1 \sqsupseteq Q_2,\ Q_2 \sqsupseteq Q_3 \vdash Q_1 \sqsupseteq Q_3$" is sound.

$\varphi(Q_1 \sqsupseteq Q_2) = \textbf{true}$ and $\varphi(Q_2 \sqsupseteq Q_3);$

$\Longrightarrow \varphi(Q_1) \supseteq \varphi(Q_2)$ and $\varphi(Q_2) \supseteq \varphi(Q_3);$

$\Longrightarrow \varphi(Q_1) \supseteq \varphi(Q_3) \Longrightarrow \varphi(Q_1 \sqsupseteq Q_3) = \textbf{true}.$

Show that "$Q_1 \sqsupseteq Q_2 \vdash {Q_2}^{\sim} \sqsupseteq {Q_1}^{\sim}$" is sound.

observe that ((Q is [every s] $\Rightarrow Q^{\sim}$ is [some s]) and (Q is [some s] $\Rightarrow Q^{\sim}$ is [every s]));

$\Longrightarrow \varphi(Q^{\sim}) = \varphi(Q)^{\sim};$

therefore, $\varphi(Q_1 \sqsupseteq Q_2) = \textbf{true} \Longrightarrow \varphi(Q_1) \supseteq \varphi(Q_2) \Longrightarrow \varphi(Q_2)^{\sim} \supseteq \varphi(Q_1)^{\sim};$

$\Longrightarrow \varphi({Q_2}^{\sim} \sqsupseteq {Q_1}^{\sim}) = \textbf{true}.$

Show that "$Q_1 \sqsupseteq Q_2 \vdash [\text{some } \langle R\ Q_1\rangle] \sqsupseteq [\text{some } \langle R\ Q_2\rangle]$" is sound.

$\varphi(Q_1 \sqsupseteq Q_2) = \textbf{true} \Longrightarrow \varphi(Q_1) \supseteq \varphi(Q_2);$

$\Longrightarrow \mathcal{RQ}_{\varphi(R)}(\varphi(Q_1)) \supseteq \mathcal{RQ}_{\varphi(R)}(\varphi(Q_2));$

$\Longrightarrow \textbf{some}(\mathcal{RQ}_{\varphi(R)}(\varphi(Q_1))) \supseteq \textbf{some}(\mathcal{RQ}_{\varphi(R)}(\varphi(Q_2)));$

$\Longrightarrow \varphi([\text{some } \langle R\ Q_1\rangle] \sqsupseteq [\text{some } \langle R\ Q_2\rangle]) = \textbf{true}.$

Show that "$Q_1 \sqsupseteq Q_2,\ Q_1 \not\sqsupseteq Q_2 \vdash \textbf{false}$" is sound.

$\varphi(Q_1 \sqsupseteq Q_2) = \textbf{true} \Longrightarrow \varphi(Q_1) \supseteq \varphi(Q_2);$

and $\varphi(Q_1 \not\sqsupseteq Q_2) = \textbf{true} \Longrightarrow \varphi(Q_1) \not\supseteq \varphi(Q_2);$

$\Longrightarrow$ contradiction $\Longrightarrow \varphi(\textbf{false}) = \textbf{true}.$

Show that "$[\text{some } \langle R\ [\text{some } s]\rangle] \sqsupseteq [\text{every } \langle R\ [\text{some } s]\rangle] \vdash [\text{some } s] \sqsupseteq [\text{every } s]$" is sound.

$\varphi([\text{some } \langle R\ [\text{some } s]\rangle] \sqsupseteq [\text{every } \langle R\ [\text{some } s]\rangle]) = \textbf{true};$

$\Longrightarrow \textbf{some}(\mathcal{RQ}_{\varphi(R)}(\textbf{some}(\varphi(s)))) \supseteq \textbf{every}(\mathcal{RQ}_{\varphi(R)}(\textbf{some}(\varphi(s))));$

$\Longrightarrow \mathcal{RQ}_{\varphi(R)}(\textbf{some}(\varphi(s))) \neq \emptyset;$

$\Longrightarrow \textbf{some}(\varphi(s)) \neq \emptyset$ since $\mathcal{RQ}_{\varphi(R)}(\emptyset) = \emptyset;$

$\Longrightarrow \varphi(s) \neq \emptyset$ since $\mathbf{some}(\emptyset) = \emptyset$;

$\Longrightarrow \mathbf{some}(\varphi(s)) \supseteq \mathbf{every}(\varphi(s))$;

$\Longrightarrow \varphi([\text{some } s] \sqsupseteq [\text{every } s]) = \mathbf{true}$.

Show that "[every r] $\not\sqsupseteq$ [every t] $\vdash$ [some r] $\sqsupseteq$ [every r]" is sound.

$\varphi([\text{every } r] \not\sqsupseteq [\text{every } t]) = \mathbf{true} \Longrightarrow \mathbf{every}(\varphi(r)) \not\supseteq \mathbf{every}(\varphi(t))$;

$\Longrightarrow \varphi(r) \not\subseteq \varphi(t) \Longrightarrow \varphi(r) \neq \emptyset$;

$\Longrightarrow \mathbf{some}(\varphi(r)) \supseteq \mathbf{every}(\varphi(r)) \Longrightarrow \varphi([\text{some } r] \sqsupseteq [\text{every } r]) = \mathbf{true}$.

Show that "[some s] $\not\sqsupseteq$ [every s] $\vdash$ [every t] $\sqsupseteq$ [every $\langle R$ [every s]$\rangle$]" is sound.

$\varphi([\text{some } s] \not\sqsupseteq [\text{every } s]) = \mathbf{true} \Longrightarrow \mathbf{some}(\varphi(s)) \not\supseteq \mathbf{every}(\varphi(s)) \Longrightarrow \varphi(s) = \emptyset$;

$\Longrightarrow \mathbf{every}(\varphi(s)) = \mathbf{every}(\emptyset) = \mathcal{P}ow(\mathcal{U})$;

$\Longrightarrow \mathcal{RQ}_{\varphi(R)}(\mathbf{every}(\varphi(s))) = \mathcal{RQ}_{\varphi(R)}(\mathcal{P}ow(\mathcal{U})) = \mathcal{U}$;

$\Longrightarrow \varphi(t) \subseteq \mathcal{RQ}_{\varphi(R)}(\mathbf{every}(\varphi(s)))$;

$\Longrightarrow \mathbf{every}(\varphi(t)) \supseteq \mathbf{every}(\mathcal{RQ}_{\varphi(R)}(\mathbf{every}(\varphi(s))))$;

$\Longrightarrow \varphi([\text{every } t] \sqsupseteq [\text{every } \langle R \text{ [every } s]\rangle]) = \mathbf{true}$.

B.3 Subsumption of the Montagovian inference rules by the BGQ logic

The last step is to show that the inference rules of the BGQ logic locally subsume those of the Montagovian syntax.

$$\frac{(\text{every } s\ t)}{(\text{every } (R\ (\text{some } s))\ (R\ (\text{some } t)))}$$

$\xrightarrow{\text{maps to}}$ [every s] $\sqsupseteq$ [every t]
$\vdash$ [some t] $\sqsupseteq$ [some s]
$\vdash$ [some $\langle R$ [some t]$\rangle$] $\sqsupseteq$ [some $\langle R$ [some s]$\rangle$]
$\vdash$ [every $\langle R$ [some s]$\rangle$] $\sqsupseteq$ [every $\langle R$ [some t]$\rangle$]
$\xrightarrow{\text{maps to}}$ (every $(R$ (some $s))$ $(R$ (some $t)))$

$$\frac{(\text{every } s\ t)}{(\text{every } (R\ (\text{every } t))\ (R\ (\text{every } s)))}$$

$\xrightarrow{\text{maps to}}$ [every s] $\sqsupseteq$ [every t]
$\vdash$ [some t] $\sqsupseteq$ [some s]
$\vdash$ [some $\langle R$ [some t]$\rangle$] $\sqsupseteq$ [some $\langle R$ [some s]$\rangle$]
$\vdash$ [every $\langle R$ [some s]$\rangle$] $\sqsupseteq$ [every $\langle R$ [some t]$\rangle$]
$\xrightarrow{\text{maps to}}$ (every $(R$ (some $s))$ $(R$ (some $t)))$

$$\frac{(\text{every } r\ s),\ (\text{every } s\ t)}{(\text{every } r\ t)}$$

$\xrightarrow{\text{maps to}}$ $[\text{every } r] \sqsupseteq [\text{every } s]$
$\xrightarrow{\text{maps to}}$ $[\text{every } s] \sqsupseteq [\text{every } t]$
$\vdash [\text{every } r] \sqsupseteq [\text{every } t]$
$\xrightarrow{\text{maps to}}$ $(\text{every } r\ t)$

$(\text{every } t\ t)$

$\vdash [\text{every } t] \sqsupseteq [\text{every } t]$
$\xrightarrow{\text{maps to}}$ $(\text{every } t\ t)$

$\exists c$

$\vdash [\text{some } c] \sqsupseteq [\text{every } c]$
$\xrightarrow{\text{maps to}}$ $\exists c$

$(\text{at-most-one } c)$

$\vdash [\text{every } c] \sqsupseteq [\text{some } c]$
$\xrightarrow{\text{maps to}}$ $(\text{at-most-one } c)$

$$\frac{\exists(R\ (\text{some } s))}{\exists s}$$

$\xrightarrow{\text{maps to}}$ $[\text{some } \langle R\ [\text{some } s]\rangle] \sqsupseteq [\text{every } \langle R\ [\text{some } s]\rangle]$
$\vdash [\text{some } s] \sqsupseteq [\text{every } s]$
$\xrightarrow{\text{maps to}}$ $\exists s$

$$\frac{\exists r,\ (\text{every } r\ t)}{\exists t}$$

$\xrightarrow{\text{maps to}}$ $[\text{some } r] \sqsupseteq [\text{every } r]$
$\xrightarrow{\text{maps to}}$ $[\text{every } r] \sqsupseteq [\text{every } t]$
$\vdash [\text{some } t] \sqsupseteq [\text{some } r]$
$\vdash [\text{some } t] \sqsupseteq [\text{some } r] \sqsupseteq [\text{every } r] \sqsupseteq [\text{every } t]$
$\vdash [\text{some } t] \sqsupseteq [\text{every } t]$
$\xrightarrow{\text{maps to}}$ $\exists t$

$$\frac{(\text{at-most-one } t),\ (\text{every } r\ t)}{(\text{at-most-one } r)}$$

$\xrightarrow{\text{maps to}}$ $[\text{every } t] \sqsupseteq [\text{some } t]$
$\xrightarrow{\text{maps to}}$ $[\text{every } r] \sqsupseteq [\text{every } t]$
$\vdash [\text{some } t] \sqsupseteq [\text{some } r]$
$\vdash [\text{every } r] \sqsupseteq [\text{every } t] \sqsupseteq [\text{some } t] \sqsupseteq [\text{some } r]$
$\vdash [\text{every } r] \sqsupseteq [\text{some } r]$
$\xrightarrow{\text{maps to}}$ $(\text{at-most-one } r)$

$$\frac{\neg(\text{every } r\ t)}{\exists r}$$

$\xrightarrow{\text{maps to}}$ $[\text{every } r] \not\sqsupseteq [\text{every } t]$
$\vdash [\text{some } r] \sqsupseteq [\text{every } r]$
$\xrightarrow{\text{maps to}}$ $\exists r$

$$\frac{\exists s,\ (\text{at-most-one}\ t),\ (\text{every}\ s\ t)}{(\text{every}\ t\ s)}$$

$\xrightarrow{\text{maps to}}$ [some s] $\sqsupseteq$ [every s]

$\xrightarrow{\text{maps to}}$ [every t] $\sqsupseteq$ [some t]

$\xrightarrow{\text{maps to}}$ [every s] $\sqsupseteq$ [every t]

$\vdash$ [some s] $\sqsupseteq$ [every s] $\sqsupseteq$ [every t] $\sqsupseteq$ [some t]

$\vdash$ [some s] $\sqsupseteq$ [some t]

$\vdash$ [every t] $\sqsupseteq$ [every s]

$\xrightarrow{\text{maps to}}$ (every t s)

$$\frac{\exists r,\ (\text{every}\ r\ s),\ (\text{every}\ r\ t)}{(\text{every}\ (R\ (\text{every}\ s))\ (R\ (\text{some}\ t)))}$$

$\xrightarrow{\text{maps to}}$ [some r] $\sqsupseteq$ [every r]

$\xrightarrow{\text{maps to}}$ [every r] $\sqsupseteq$ [every s]

$\xrightarrow{\text{maps to}}$ [every r] $\sqsupseteq$ [every t]

$\vdash$ [some t] $\sqsupseteq$ [some r]

$\vdash$ [some t] $\sqsupseteq$ [some r] $\sqsupseteq$ [every r] $\sqsupseteq$ [every s]

$\vdash$ [some t] $\sqsupseteq$ [every s]

$\vdash$ [some $\langle R$ [some t]$\rangle$] $\sqsupseteq$ [some $\langle R$ [every s]$\rangle$]

$\vdash$ [every $\langle R$ [every s]$\rangle$] $\sqsupseteq$ [every $\langle R$ [some t]$\rangle$]

$\xrightarrow{\text{maps to}}$ (every (R (every s)) (R (some t)))

$$\frac{\neg\exists s}{(\text{every}\ t\ (R\ (\text{every}\ s)))}$$

$\xrightarrow{\text{maps to}}$ [some s] $\not\sqsupseteq$ [every s]

$\vdash$ [every t] $\sqsupseteq$ [every $\langle R$ [every s]$\rangle$]

$\xrightarrow{\text{maps to}}$ (every t (R (every s)))

$$\frac{(\text{at-most-one}\ t),\ (\text{every}\ s\ t)}{(\text{every}\ (R\ (\text{some}\ s))\ (R\ (\text{every}\ t)))}$$

$\xrightarrow{\text{maps to}}$ [every t] $\sqsupseteq$ [some t]

$\xrightarrow{\text{maps to}}$ [every s] $\sqsupseteq$ [every t]

$\vdash$ [every s] $\sqsupseteq$ [some t]

$\vdash$ [every t] $\sqsupseteq$ [some s]

$\vdash$ [some $\langle R$ [every t]$\rangle$] $\sqsupseteq$ [some $\langle R$ [some s]$\rangle$]

$\vdash$ [every $\langle R$ [some s]$\rangle$] $\sqsupseteq$ [every $\langle R$ [every t]$\rangle$]

$\xrightarrow{\text{maps to}}$ (every (R (some s)) (R (every t)))

$$\frac{\exists r,\ (\text{every } r\ s),\ (\text{every } r\ t)}{(\text{some } s\ t)}$$

$\xrightarrow{\text{maps to}}$ $[\text{some } r] \sqsupseteq [\text{every } r]$

$\xrightarrow{\text{maps to}}$ $[\text{every } r] \sqsupseteq [\text{every } s]$

$\xrightarrow{\text{maps to}}$ $[\text{every } r] \sqsupseteq [\text{every } t]$

$\vdash [\text{some } s] \sqsupseteq [\text{some } r]$

$\vdash [\text{some } s] \sqsupseteq [\text{some } r] \sqsupseteq [\text{every } r] \sqsupseteq [\text{every } t]$

$\vdash [\text{some } s] \sqsupseteq [\text{every } t]$

$\xrightarrow{\text{maps to}}$ $(\text{some } s\ t)$

$$\frac{\Psi,\ \neg\Psi}{\textbf{false}}$$

$\xrightarrow{\text{maps to}}$ $Q_1 \sqsupseteq Q_2\ (Q_1 \not\sqsupseteq Q_2)$

$\xrightarrow{\text{maps to}}$ $Q_1 \not\sqsupseteq Q_2\ (Q_1 \sqsupseteq Q_2)$

$\vdash \textbf{false}$

$\xrightarrow{\text{maps to}}$ $\textbf{false}$

Appendix C

Semantic Interpretation

C.1 Syntactic and semantic rules

SENTENCES

Complete sentences:

(1)	$\mathrm{S} \leftarrow \mathrm{S}'$	$\mathrm{S}'_d = \mathcal{U} \times \mathcal{U}$
(2)	$\mathrm{S} \leftarrow$ IT–IS–TRUE S'	$\mathrm{S}'_d = \mathcal{U} \times \mathcal{U}$
(3)	$\mathrm{S} \leftarrow$ IT–IS–NOT–TRUE S'	$\mathrm{S}'_d = \emptyset$

Orientation of a sentence with a virtual NP:

(4)	$\mathrm{S}^* \leftarrow \mathrm{S}'$	$\mathrm{S}^*_d = \mathrm{S}'_d$
(5)	$\mathrm{S}^* \leftarrow \mathrm{S}'$	$\mathrm{S}^*_d = {\mathrm{S}'_d}^{-1}$

Unbound sentences:

(6)	$\mathrm{S}' \leftarrow$ NP VP	$\mathrm{S}'_d = \mathrm{NP}_{\mathrm{after}},\ \mathrm{NP}_{\mathrm{before}} = {\mathrm{VP}_d}^{-1}$
(7)	$\mathrm{S}' \leftarrow$ NP NOT VP	$\mathrm{S}'_d = \overline{\mathrm{NP}}_{\mathrm{after}},\ \mathrm{NP}_{\mathrm{before}} = {\mathrm{VP}_d}^{-1}$
(8)	$\mathrm{S}' \leftarrow$ NP NOT VP	$\mathrm{S}'_d = \mathrm{NP}_{\mathrm{after}},\ \mathrm{NP}_{\mathrm{before}} = {\overline{\mathrm{VP}}_d}^{-1}$

VERB PHRASES

Ordinary transitive verbs:

(9)	$\mathrm{VP} \leftarrow$ V NP	$\mathrm{VP}_d = \mathrm{NP}_{\mathrm{after}},\ \mathrm{NP}_{\mathrm{before}} = \mathrm{V}_d$

Verb *'to be'*:

(10) VP $\leftarrow$ BE NP $\qquad$ $\mathrm{VP}_d = \mathrm{NP}_{\mathrm{after}},\ \mathrm{NP}_{\mathrm{before}} = I$

(11) VP $\leftarrow$ BE ADJ $\qquad$ $\mathrm{VP}_d = (\mathrm{ADJ}_d \times \mathcal{U})$

Verb *'to have'*:

(12) VP $\leftarrow$ HAVE NP $\qquad$ $\mathrm{VP}_d = \mathrm{NP}_{\mathrm{after}},\ \mathrm{NP}_{\mathrm{before}} = \mathtt{possess}$

(13) VP $\leftarrow$ HAVE NP $\qquad$ $\mathrm{VP}_d = \mathrm{NP}_{\mathrm{after}}{}^{-1},\ \mathrm{NP}_{\mathrm{before}} = \mathcal{U} \times \mathcal{U}$

(14) VP $\leftarrow$ HAVE NP AS R $\qquad$ $\mathrm{VP}_d = \mathrm{NP}_{\mathrm{after}},\ \mathrm{NP}_{\mathrm{before}} = \mathrm{R}_d$

Noun Phrases

Reflexive pronouns:

(15) NP $\leftarrow$ XSELF $\qquad$ $\mathrm{NP}_{\mathrm{after}} = \Phi(\mathrm{NP}_{\mathrm{before}})$

Optional binding of the floating variable by the NP:

(16) NP $\leftarrow$ $\mathrm{NP}^{(0)}$ $\qquad$ $\mathrm{NP}^{(0)}_{\mathrm{before}} = \Phi(\mathrm{NP}_{\mathrm{before}})^{-1}$

(17) NP $\leftarrow$ $\mathrm{NP}^{(0)}$

Virtual NP of a relative clause:

(18) $\mathrm{NP}^{(0)} \leftarrow \perp$ $\qquad$ $\mathrm{NP}^{(0)}_{\mathrm{after}} = \mathrm{NP}^{(0)}_{\mathrm{before}}$

Quantification of the NP:

(19) $\mathrm{NP}^{(0)} \leftarrow$ DET NP'' $\qquad$ $\mathrm{NP}^{(0)}_{\mathrm{after}} = \mathrm{NP}''_{\mathrm{out}},\ \mathrm{NP}''_{\mathrm{in}} = \mathrm{NP}^{(0)}_{\mathrm{before}} \circ_{\mathrm{DET}} \mathrm{NP}''_r$

(20) $\mathrm{NP}^{(0)} \leftarrow \mathrm{NP}''$ $\qquad$ $\mathrm{NP}^{(0)}_{\mathrm{after}} = \mathrm{NP}''_{\mathrm{out}},\ \mathrm{NP}''_{\mathrm{in}} = \mathrm{NP}^{(0)}_{\mathrm{before}} \circ_{\mathrm{some}} \mathrm{NP}''_r$

(21) $\mathrm{NP}^{(0)} \leftarrow$ DET OF NP' $\qquad$ $\mathrm{NP}^{(0)}_{\mathrm{after}} = \mathrm{NP}'_{\mathrm{out}},\ \mathrm{NP}'_{\mathrm{in}} = \mathrm{NP}^{(0)}_{\mathrm{before}} \circ_{\mathrm{DET}} \mathrm{NP}'_r$

(22) $\mathrm{NP}^{(0)} \leftarrow \mathrm{NP}'$ $\qquad$ $\mathrm{NP}^{(0)}_{\mathrm{after}} = \mathrm{NP}'_{\mathrm{out}},\ \mathrm{NP}'_{\mathrm{in}} = \mathrm{NP}^{(0)}_{\mathrm{before}} \circ_{\mathrm{the}} \mathrm{NP}'_r$

Definites:

(23) $\mathrm{NP}' \leftarrow$ PRONOUN $\qquad$ $\mathrm{NP}'_{\mathrm{out}} = \mathrm{NP}'_{\mathrm{in}},\ \mathrm{NP}'_r = \mathbf{referent}$

(24) $\mathrm{NP}' \leftarrow$ THE NP'' $\qquad$ $\mathrm{NP}'_r = \mathbf{referent}(\mathrm{NP}''_r)$

(25) $\mathrm{NP}' \leftarrow$ POSS NP'' $\qquad$ $\mathrm{NP}'_{\mathrm{out}} = \mathrm{POSS}_{\mathrm{out}},\ \mathrm{POSS}_{\mathrm{in}} = \mathrm{NP}'_{\mathrm{in}}$

Integration of the optional word *'other'* in the NP:

(26) $\mathrm{NP}'' \leftarrow$ OTHER $\mathrm{NP}^{(3)}$ $\qquad$ $\mathrm{NP}''_r = \mathrm{NP}^{(3)}_r \cap \overline{\mathbf{referent}}$

(27) $\quad NP'' \leftarrow NP^{(3)}$

Integration of an optional relative clause in the NP:

(28) $\quad NP^{(3)} \leftarrow NP^{(4)}\ REL \qquad NP^{(3)}_r = NP^{(4)}_r \cap REL_d$

(29) $\quad NP^{(3)} \leftarrow NP^{(4)}$

Integration of an optional list of adjectives in the NP:

(30) $\quad NP^{(4)} \leftarrow ADJ\ NP^{(4)} \qquad NP^{(4)}_r \leftarrow NP^{(4)}_r \cap (ADJ_d \times \mathcal{U})$

(31) $\quad NP^{(4)} \leftarrow NP^{(5)}$

Integration of an optional 'of' prepositional phrase in the NP:

(32) $\quad NP^{(5)} \leftarrow NP^{(6)}\ OF\ NP \qquad NP^{(5)}_r = NP^{(6)}_r,\ NP^{(5)}_{out} = NP_{after},\ NP_{before} = NP^{(5)}_{in}$

(33) $\quad NP^{(5)} \leftarrow NP^{(6)}\ OF\ NP \qquad NP^{(5)}_{out} = NP^{(5)}_{in},\ NP^{(5)}_r = NP_{after},\ NP_{before} = NP^{(6)}_r$

(34) $\quad NP^{(5)} \leftarrow NP^{(6)} \qquad NP^{(5)}_{out} = NP^{(5)}_{in}$

Integration of the common noun in the NP:

(35) $\quad NP^{(6)} \leftarrow CN \qquad NP^{(6)}_r = CN_d$

Common Nouns

(36) $\quad CN \leftarrow N \qquad CN_d = N_d \times \mathcal{U}$

(37) $\quad CN \leftarrow N \qquad CN_d = (N_d \times \mathcal{U}) \cap \mathtt{possess}^{-1}$

(38) $\quad CN \leftarrow R \qquad CN_d = R_d$

(39) $\quad CN \leftarrow R \qquad CN_d = (\mathrm{dom}(R_d) \times \mathcal{U})$

(40) $\quad CN \leftarrow R \qquad CN_d = (\mathrm{dom}(R_d) \times \mathcal{U}) \cap \mathtt{possess}^{-1}$

Relative Clauses

(41) $\quad REL \leftarrow PREL\ S^* \qquad REL_d = S^*_d$

(42) $\quad REL \leftarrow WHOSE\ NP''\ S^* \qquad REL_d = NP''^{\,-1}_{out},\ NP''_{in} = S^*_d \circ_{the} NP''_r$

(43) $\quad REL \leftarrow WHOSE\ NP''\ S^* \qquad REL_d = \Phi(NP''_{out}),\ NP''_{in} = S^*_d \circ_{the} NP''_r$

Possessives

(44) $\quad POSS \leftarrow NP\ \ \text{'s} \qquad POSS_{out} = NP_{after},\ NP_{before} = POSS_{in}$

(45) $\text{POSS} \leftarrow \text{POSSADJ}$ $\quad\quad \text{POSS}_{\text{out}} = \text{POSS}_{\text{in}} \circ_{\text{the}} \textbf{referent}$

TERMINAL SYMBOLS.

(1) POSSADJ $\leftarrow$ my | your | his | her | its | our | their

(2) PRONOUN $\leftarrow$ I | me | you | he | him | she | her | it | we | us | they | them

(3) XSELF $\leftarrow$ myself | yourself | himself | herself | itself | ourselves | yourselves | themselves

(4) PREL $\leftarrow$ that | who | whom | which | $\perp$

(5) DET $\leftarrow$ some | every | no | most | at-least k | at-most k | exactly k

Appendix D

Properties: Abstract Generalized Quantifiers

The study of inference in the AGQ logic is left for future work. In this section, we present some basic properties of the logic.

Contrary to the extended relation algebra, there is no need for an inverse operator, since it is computable by the conditional inclusion.

$$\textbf{some}(I) \quad \supseteq_{R^{-1}} \quad \textbf{every}(R). \tag{D.1}$$

These two rules represent, the analog, for the conditional inclusion, of the reflexivity rule and the transitivity rule.

$$Q \supseteq_R Q \quad \Longrightarrow \quad R \supseteq I; \tag{D.2}$$

$$Q_1 \supseteq_{R_1} Q_2,\ Q_2 \supseteq_{R_2} Q_3 \quad \Longrightarrow \quad \exists R \supseteq R_1 \circ R_2,\ Q_1 \supseteq_R Q_3. \tag{D.3}$$

The dual operation behaves as usual.

$$Q_1 \supseteq_R Q_2 \quad \Longrightarrow \quad Q_2{}^{\sim} \supseteq_{R^{-1}} Q_1{}^{\sim}. \tag{D.4}$$

The following three rules give the behavior of the conditional inclusion with respect to the intersection and the union of abstract generalized quantifiers. (For the intersection and the union on the right-hand side, just take the dual of these rules.)

$$Q_1 \supseteq_{R_1} Q,\ Q_2 \supseteq_{R_2} Q \quad \Longrightarrow \quad Q_1 \cup Q_2 \supseteq_{R_1 \cup R_2} Q; \tag{D.5}$$

$$Q_1 \supseteq_{R_1} Q,\ Q_2 \supseteq_{R_2} Q \quad \Longrightarrow \quad \exists R \supseteq R_1 \cap R_2,\ Q_1 \cap Q_2 \supseteq_R Q; \tag{D.6}$$

$$(\text{D.7}) \qquad Q_1 \supseteq_{R_1} \mathbf{every}(R), Q_2 \supseteq_{R_2} \mathbf{every}(R) \implies Q_1 \cap Q_2 \supseteq_{R_1 \cap R_2} \mathbf{every}(R).$$

Bibliography

Barwise, J. and Cooper, R. (1981). Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4:159–219.

Benthem, J. v. (1984). Questions about quantifiers. *The Journal of Symbolic Logic*, 49(2):443–466.

Borgida, A. and Patel-Schneider, P. F. (1994). A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:277–308.

Böttner, M. (1992). Variable-free semantics for anaphora. *Journal of Philosophical Logic*, 21:375–390.

Does, J. v. d. (1991). A generalized quantifier logic for naked infinitives. *Linguistics and Philosophy*, 14:241–294.

Givan, R., McAllester, D., and Shalaby, S. (1991). Natural language based inference procedures applied to Schubert's steamroller. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI–91)*, pages 915–920.

Heim, I. (1982). *The Semantics of Definite and Indefinite Noun Phrases*. PhD thesis, University of Massachusetts at Amherst.

Hirst, G. (1981). *Anaphora in Natural Language Understanding: A Survey*. Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York.

Kadmon, N. (1987). *On Unique and Non-Unique Reference and Asymmetric Quantification*. PhD thesis, University of Massachusetts at Amherst.

Kamp, H. (1981). A theory of truth and semantic representation. In Groenendijk, J., Janssen, T., and Stokhof, M., editors, *Formal Methods in the Study of Language*. Mathematical Center, Amsterdam.

Levesque, H. J. and Brachman, R. J. (1987). Expressiveness and tractability in knowledge representation. *Computational Intelligence*, 3(2):78–93.

McAllester, D. (1993). Automatic recognition of tractability in inference relations. *Journal of the Association for Computing Machinery*, 40(2):284–303.

McAllester, D., Givan, B., and Fatima, T. (1989). Taxonomic syntax for first order inference. In *Proceedings First International Conference on Principles of Knowledge Representation and Reasoning*, pages 289–300, Toronto, Ont.

McAllester, D. A. and Givan, R. (1992). Natural language syntax and first-order inference. *Artificial Intelligence*, 56:1–20.

McAllester, D. A. and Givan, R. (1993). Taxonomic syntax for first order inference. *Journal of the Association for Computing Machinery*, 40(2):246–283.

Montague, R. (1974). *Formal philosophy: selected papers of Richard Montague*. Yale University Press.

Moore, R. (1981). Problems in logical form. In *Proceedings, 19th Annual Meeting of the Association for Computational Linguistics*, pages 117–124, Stanford, California.

Partee, B. H., Meulen, A. t., and Wall, R. E. (1993). *Mathematical Methods in Linguistics*, chapter 14, pages 371–399. Kluwer Academic Publishers, Dordrecht / Boston / London, corrected version of the first edition edition.

Patel-Schneider, P. F. (1989). Undecidability of subsumption in NIKL. *Artificial Intelligence*, 39(2):263–272.

Purdy, W. C. (1991). A logic for natural language. *Notre Dame Journal of Formal Logic*, 32(3):409–425.

Purdy, W. C. (1992a). Surface reasoning. *Notre Dame Journal of Formal Logic*, 33(1):13–36.

Purdy, W. C. (1992b). A variable-free logic for mass terms. *Notre Dame Journal of Formal Logic*, 33(3):348–359.

Suppes, P. (1976). Elimination of quantifiers in the semantics of natural language by use of extended relation algebras. *Revue Internationale de Philosophie*, 30:243–259.

Westerståhl, D. (1984). Some results on quantifiers. *Notre Dame Journal of Formal Logic*, 25(2):152–170.