LOGICAL ANALYSIS OF FRAGMENTS OF NATURAL LANGUAGE

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Engineering and Physical Sciences

2006

By
Allan Third
School of Computer Science

Contents

Approximate Word Count = 50 145

List of Tables

List of Figures

Abstract

Let a fragment of a natural language be a set of sentences of that language, selected according to the syntactic constructions they exhibit. Suppose that each sentence in a fragment can be assigned a semantic representation in some logical formalism. We can treat such a fragment as defining a fragment of the relevant logical formalism, and ask questions regarding the expressive power and reasoning capabilities of that logical fragment. The results can then be interpreted as describing the semantic properties of the original natural language fragment.

In this thesis, we define several fragments of English containing features such as simple noun phrases, relative clauses, verbs and coordination. Each of the fragments we consider can be given semantics in first-order logic. By making use of standard results and techniques of first-order proof- and model-theory, we determine, for each fragment F, the computational complexity of determining whether a given set of sentences of F is logically consistent. For selected fragments, we also characterise expressive power in terms of the ability each has to describe the differences between two situations, and use that characterisation to provide a criterion for when arbitrary expressions of first-order logic can be translated into each fragment.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.

Copyright

Acknowledgements

Cui dono lepidum novum libellum arida modo pumice expolitum?

— G. Valerius Catullus, c. DCLXIX–DXXCIX *ab urbe condita*

First and foremost, I'd like to express my enormous gratitude to my supervisor, Dr. Ian Pratt-Hartmann, without whose teaching and support (and prodding!), this thesis could never have come into being. In particular, I'd like to thank Ian for the patience he's shown me during the final stages of writing.

Thanks, and all my love, to my parents, for their fantastic support, both emotional, and financial, throughout, and to Rachael and Niall, for accommodating all the random visits, amongst everything else.

Angela, who has put up with far too much privation and inconvenience in the name of getting this thesis finished, and who has been incredible in her support nonetheless: thank you so much for everything, and all my love.

Thanks to Don for all the crosswords, Latin, Gaelic and fellow procrastination. Thanks to Moneesha for all her much-needed support (and for all the food!) Everyone who shared the office over the last few years has been great – thanks in particular to Aled and Savas for putting up with my eccentricities in the last few months.

Thanks to Caroline and Claire, for long, meandering chats, in Manchester, and at great distance. To everyone on the list, for everything from phone calls to a succession of unusual Hogmanay destinations: thanks, all of you.

The support of the EPSRC is gratefully acknowledged.

Chapter 1

Introduction

A man is seldom in a humour to unlock his bookcase, set his desk in order, and betake himself to serious study.

— Dr. Johnson

The various syntactic constructions of a natural language such as English each have a characteristic interpretation. Intuitively, proper nouns serve to identify individuals, transitive verbs express relations, and so on. Different combinations of individual syntactic constructions interact to allow the expression of more complex meanings. Is there a way to make these intuitions precise, and to characterise the effects on expressive power of combining different elements of syntax?

This thesis sets out to establish that the answer to these questions is yes, and to provide, by way of demonstration, characterisations of the expressive power of several different *fragments of English*, constructed from a small set of syntactic features in varying combinations.

In the literature on formal logic, it is relatively common to carry out such analysis of new, or fragments of existing, logical languages. Among the questions often posed of a logic $\mathcal{L}$ are: how difficult is it to decide algorithmically whether a given set of $\mathcal{L}$-formulae are consistent, if that is even possible, (the *satisfiability problem*), and what is the ability of $\mathcal{L}$ to distinguish different situations (its *expressive power*)? And yet, despite the extent to which linguists have taken to the use of formal logic to represent the semantics of natural language, there seems to have been very little work done on discussing these same problems for natural language. We aim in this thesis to contribute towards closing this gap.

There are several reasons for wanting to do so. A better understanding of the semantic interactions of syntactic processes is intrinsically linguistically interesting. It might also be instructive to be able to contrast the logical strengths and weaknesses of natural languages and the artificial languages of logicians. Finally, more practically, there are potential applications in computing. The most obvious example is that of natural language interfaces to databases, for which it is important to know that the particular fragment of English used is capable of expressing every query required. If such software has a deductive component, then the computational properties of carrying out reasoning with that fragment can also be very useful for software designers to know.

The work presented here is a continuation, and extension, of the programme proposed by Pratt-Hartmann in [58]. That paper defines a small fragment of English and several linguistically interesting extensions of it. By means of standard translations into first-order logic, each of these fragments is analysed with regard to its *semantic complexity* – that is, the complexity of deciding its satisfiability problem. It is proposed in [58] that such an approach could be extended to include a wider range of English constructions. Pratt-Hartmann and Third continue this programme in [60] with the extension of the fragments of [58] with ditransitive verbs. The present work contains some of the results presented earlier, as well as further semantic complexity results for fragments of English containing various levels of (sub)sentential coordination. Some of the fragments of [58] and [60] are also revisited, and their expressive power is characterised in terms of their ability to distinguish different situations. These results also offer criteria by which we can determine when arbitrary logical formulae can be translated back into English sentences in the fragments under consideration.

More specifically, we consider a range of fragments of English constructed using proper and common nouns, the determiners every, some and no, relative clauses, the copula, (di)transitive verbs and coordination of various sentential and subsentential constituents. For each of these fragments, we present semantic complexity results, with proofs for those which are novel. For each fragment not featuring subsentential coordination, we present an analysis of its expressive power.

The contributions made by this thesis can be summarised as follows. By continuing the programme of [58] with regard to questions of the complexity of the satisfiability problem, we contribute towards the understanding of the deductive

power of simple English constructions, and in a sense help to "map out" parts of the interface between syntax and semantics, at least for English. This goal is also aided by our analyses of the logical expressive power of the fragments we study, and by the presentation, in a linguistics context, of the techniques needed for such analysis. We show that it is not only possible, but practical, to answer these questions for natural languages using standard logical formalisms, and without needing to invent a custom "English-like" logical syntax. As a result, not only are the techniques we use quite generally applicable to non-English languages, but we are also able to take advantage of the full body of existing work on the syntax and semantics of first-order logic.

The structure of this thesis is as follows. Chapter 2 reviews the existing literature on the topics of natural language reasoning and expressive power. In Chapter 3, we give a brief summary of the necessary background topics in logic, and in Chapter 4, we give a similar summary of the relevant topics in linguistics, as well as a more detailed description of the questions we answer. Chapters 5 and 6 form the main body of the thesis: Chapter 5 gives satisfiability and expressive power results for fragments featuring the copula, relative clauses and (di)transitive verbs and Chapter 6 gives satisfiability results for each of the fragments of Chapter 5 extended in turn with various levels of (sub)sentential coordination. Finally, in Chapter 7, we summarise our results, and draw some general conclusions from them, before suggesting some questions to be addressed by future work.

Chapter 2

Related Work

Que sais-je?

— Montaigne, *Essais*

This chapter contains a brief summary of related work. We give an overview of the existing literature and discuss the differences in approach taken by this thesis.

As stated earlier, our aim is to define a range of fragments of English with associated semantics, and ask, for each, whether the problem of determining satisfiability is decidable. In certain cases, we also give a semantic characterisation of the fragment in question, by analysing its expressive power as the ability to distinguish different situations. There are therefore two bodies of literature relevant to our work: previous studies of the problem of carrying out *inference* from natural language sentences, and previous studies of the *expressive power* of natural language constructions. We consider these in turn.

2.1 Natural Language Reasoning

A great deal of the existing work on natural language semantics is concerned with finding the best means of representing or computing the truth or assertability conditions of expressions in natural language, or, given a particular choice of representation language, with working out precisely how to interpret individual constructions in an individual language. Surprisingly little, however, seems to have been done on the study of natural languages as systems of inference. We

consider the main examples here.

This section can be divided into two parts. We consider first the various attempts there have been to study natural language reasoning using non-standard logics, by which we mean formal languages or proof systems designed specifically for this task. We then consider the rather smaller body of literature discussing natural language reasoning using general-purpose standard logics, such as first-order logic.

The motivation behind the first approach seems to be unease at the gap between the syntax of, say, first-order logic and its associated proof systems, and the syntax and kinds of inference typically used to express arguments in natural languages such as English. A key proponent of this view is Sommers, who argues, in [73], against the wholesale move by logicians towards the predicate-and-bound-variable logics stemming from Frege, in favour of a "traditional formal logic" continuing the Aristotelian tradition of a subject-predicate logic. Thus, to give the classical example, the validity of the argument

Every man is mortal
Socrates is a man
Socrates is mortal

is explained by the fact that the subject – every man – of the first premise is *positive* and *universal*, and shares a term – men – with the predicate of the second premise. The *particular* subject – Socrates – of the second premise can therefore be substituted for the subject of the first to give the (valid) conclusion Socrates is mortal.

The work of Sommers is concerned with the general foundation of logic itself, which falls somewhat outside the scope of this thesis. However, many of those we now consider hold similar views on semantics to those expressed in [73] and so it bears mentioning here.

In what follows, we examine in turn the works of Fitch [15], Suppes [79], Purdy ([62], [63], [64], [65], [67]) and McAllester and Givan [46].

Fitch, in [15], tries to formalise directly the processes of reasoning with English sentences, without any need for a separate formal language to represent semantics. He does so by defining a system of "natural deduction" inference rules which apply to the constituents of a sentence. For example, the rules of any-elimination and introduction can be represented by the schemata

$$\frac{\mathsf{b\ is\ a\ c} \qquad \mathsf{\ldots any\ c\ \ldots}}{\mathsf{\ldots b\ \ldots}} \qquad \frac{\mathsf{b\ is\ a\ c} \qquad \mathsf{\ldots b\ \ldots}}{\mathsf{\ldots any\ c\ \ldots}}$$

where ... any c ... and ... b ... are sentences identical save that in the latter, the leftmost occurrence of b occurs in the position in which the leftmost occurrence of any c occurs in the former. The rule of any-introduction is also subject to the restriction that if any other proper noun is substituted throughout a proof for b, the whole proof must continue to hold. Unsurprisingly, these rules bear a strong resemblance to the usual formulations of ∀-introduction and elimination in natural deduction systems for first-order logic.

Similar rules are presented for the introduction and elimination of other determiners, such as some, no and the, passive verb constructions, verbal negation and relative clauses. Rules are also given for sentences containing modal verbs such as believe and know. For example, via rules of infinitive-introduction and elimination, the sentences Jack believes that Bill came and Jack believes Bill to have come can be shown to be equivalent. Finally, of course, natural deduction rules for the usual Boolean connectives are presented.

The main difficulty with Fitch's approach is no doubt already apparent. Without any detailed account of both the syntax and compositional semantics of the sentences considered, it is necessary to introduce separate rules for every different syntactic construction, and every variant way of phrasing a given set of truth conditions. This point is particularly driven home by the handling of relative clauses, for which Fitch gives seven pairs of introduction and elimination rules – to handle relative clauses in the scope of any, some, no, the, not every, and so on. So, for example, any-which-introduction and -elimination is performed as follows

$$\frac{\mathsf{\ldots any\ c\ which\ vis\ \ldots}}{\mathsf{if\ any\ c\ vis,\ then\ (\ldots that\ c\ \ldots)}} \qquad \frac{\mathsf{if\ any\ c\ vis,\ then\ (\ldots that\ c\ \ldots)}}{\mathsf{\ldots any\ c\ which\ vis\ \ldots}}$$

where ... any c ... and ... that c ... are interpreted as above, and vi is an intransitive verb. Similarly, not-every-which-introduction and -elimination are given by

$$\frac{\mathsf{\ldots not\ every\ c\ which\ vis\ \ldots}}{\mathsf{some\ c\ vis,\ and\ not\ (\ldots that\ c\ \ldots)}} \qquad \frac{\mathsf{some\ c\ vis,\ and\ not\ (\ldots that\ c\ \ldots)}}{\mathsf{\ldots not\ every\ c\ which\ vis\ \ldots}}$$

with the same notation.

As we see later, such a proliferation of rules is unnecessary – the semantics of relative clauses can be separated from the semantics of the determiners in whose scope they fall without any loss.

Suppes, in [79], takes a more compositional approach. He defines a fragment of English via phrase structure rules, each of which has an associated function detailing how to form the semantics of a phrase from the semantics of its constituents. The fragment he defines contains the determiners all, some, no, demonstrative phrases there are, common nouns, adjectives, relative clauses, possessive of, (in)transitive verbs and negation. Thus the following sentences belong to Suppes' fragment.

> There are vegetables
> Some people eat all vegetables
> Some people that do not eat all vegetables love vegetables of Southern growers

Each such sentence is assigned semantics using a simple language of sets and relations. Given a vocabulary of set and relation symbols interpreted in some domain of objects, the usual operations of union ($X \cup Y$), intersection ($X \cap Y$) and complement ($\overline{X}$) are augmented with the *converse* ($\breve{R}$) of a relation R, the *restriction* ($R_{|A}$) of the domain of a relation R to a set A and the *image* ($R"A$) of a set A under the relation R. Suppes also introduces a method of fixing an *unspecified* relation between two sets, which he uses to interpret "possessive of", as in vegetables of Southern growers. If X and Y are set symbols, then $X_2Y \subseteq X \times Y$ is a relation. No formal definition is given, but we assume that the actual denotation of each X_2Y over a given domain is fixed simultaneously with the denotations of X and Y themselves.

Every common noun, and every adjective, is treated as a set symbol, and each transitive verb is treated as a relation symbol. The above sentences are then given the semantics

$$vegetables \neq \emptyset$$
$$people \cap (\cap_{v \in vegetables}(\breve{eat}"\{v\})) \neq \emptyset$$
$$(people \cap ((\neg \breve{eat})) \cap (\breve{love}"(vegetables_2"(southern \cap growers)))) \neq \emptyset.$$

The first of these presumably requires no explanatory gloss. The second can be paraphrased as stating that the intersection of the set of people and the set of those who are eaters of every vegetable is non-empty. The third example states that there exists at least one entity in the domain of discourse which is a person,

to which at least one vegetable stands in the not-eaten-by relation and which is in the love relation with every vegetable related to something which is both Southern and a grower.

Such translations do seem to represent what most native speakers would take to be the truth conditions of the given sentences. However, Suppes does not go on to use these translations directly as a tool for reasoning. Rather, he proceeds much in the manner of the traditional logician, by categorising sentences into classes and then listing valid inferences in terms of those classes, using the set-theoretic translations only to justify claims of validity. For example, all phrases which are interpreted by the subset relation are identified. They are

$$\begin{aligned}&\mathsf{all\ N_1\ are\ N_2}\\ &\mathsf{N_1}\text{ is of the form }\mathsf{adjective\ N_2}\\ &\mathsf{N_1}\text{ is of the form }\mathsf{N_2\ that\ VP}\\ &\mathsf{N_1}\text{ is of the form }\mathsf{N_2\ of\ N_3}.\end{aligned}$$

where N_i is a noun or noun phrase for $1 \leq i \leq 3$ and VP is a verb phrase. With every phrase of the above forms denoted $C(\mathsf{N_1},\mathsf{N_2})$, Suppes presents inference rules such as

$$\frac{\mathsf{All\ N_1\ VP} \quad C(\mathsf{N_1},\mathsf{N_2})}{\mathsf{All\ N_2\ VP}}$$

and

$$\frac{\mathsf{Some\ N_1\ VP} \quad C(\mathsf{N_1},\mathsf{N_2})}{\mathsf{Some\ N_2\ VP.}}$$

The soundness of such rules are justified by appeal to the set-theoretic semantics.

A similar criticism applies to this approach as to that of Fitch. By stating rules of inference directly in terms of (sets of) expressions of English, Suppes can be forced, when faced with some extension of his fragment, to change his system of inference in order to accommodate the extensions. Such a lack of extensibility is particularly odd given how straightforward it is likely to be to define a proof system which operates directly on the set-theoretic semantic representation language he defines, and which would apply unaltered to the semantics of a wide range of English constructions.

In several papers ([62], [63], [64], [65], [67]), Purdy proposes a logical formalism, $\mathcal{L}_N$, which he claims is particularly suited to represent reasoning carried

out in natural language. The advantages claimed for $\mathcal{L}_N$ are, first, that carrying out deduction in it is similar to the process of human reasoning in English, a process which (Purdy claims) operates on constructions of English syntax. Second, although satisfiability in $\mathcal{L}_N$ itself is undecidable, it has a subfragment, very similar to Quine's *fluted fragment* of first-order logic, into which a reasonably naturally-delineated fragment of English can be translated, and satisfiability in the fluted fragment is decidable in non-deterministic exponential time. Decidability is desirable for Purdy, not just on computational grounds, but also because he conjectures that "everyday" English is decidable, and therefore its semantics ought to be represented in a decidable formalism.

Formulae of $\mathcal{L}_N$ are formed from a vocabulary of n-ary predicate symbols. The following definition gives only that decidable fragment of $\mathcal{L}_N$ used to represent the semantics of English.

Definition 2.1. The $\mathcal{L}_N$ expressions of arity n are defined as follows.

1. Every n-ary predicate is an n-ary expression.
2. The complement $\overline{X}$ of an n-ary expression X is n-ary.
3. If X is n-ary and Y is m-ary, then $X \cap Y$ is $\max\{n, m\}$-ary.
4. If X is n-ary and Y is m-ary, then $X \circ Y$ is $((n + m) - 1)$-ary.
5. If X is unary and Y is $(n + 1)$-ary, then $\mathsf{some}XY$ is n-ary.

Let D be a domain of objects and $\mathcal{I}$ an *interpretation function* mapping every n-ary predicate to a subset of D^n. An n-ary expression X is satisfied by an ordered sequence $\alpha = \langle d_1, \ldots, d_n \rangle$ of elements of D, written $\alpha \models_{\mathcal{I}} X$, if

1. X is an n-ary predicate and $(d_1, \ldots, d_n) \in \mathcal{I}(X)$,
2. X is of the form $\overline{Y}$ and $\alpha \not\models_{\mathcal{I}} Y$,
3. X is of the form $Y \cap Z$ and $\alpha \models_{\mathcal{I}} Y$ and $\alpha \models_{\mathcal{I}} Z$,
4. X is of the form $Y \circ Z$, where Y is m-ary, Z is $((n + 1) - m)$-ary and for every $\beta = \langle e_1, \ldots, e_m \rangle$ such that $\beta \models_{\mathcal{I}} Y$ and $\gamma = \langle f_1, \ldots, f_{(n+1)-m} \rangle$ such that $\gamma \models_{\mathcal{I}} Z$, if $e_m = f_1$, then $\langle e_1, \ldots, e_m, f_2, \ldots f_{(n+1)-m} \rangle \models_{\mathcal{I}} X$.

5. X is of the form someYZ, where Y is unary and Z is $(n+1)$-ary, and for some $d \in D$, $\langle d \rangle \models_{\mathcal{I}} Y$ and $\langle d, d_1, \ldots, d_n \rangle \models_{\mathcal{I}} Z$.

For any pair of expressions X, Y, the expressions $X \cup Y, X \subseteq Y$ and $X \equiv Y$ are defined as abbreviations for $\overline{(\overline{X} \cap \overline{Y})}$, $\overline{(X \cap \overline{Y})}$ and $(X \subseteq Y) \cap (Y \subseteq X)$, respectively. If X is unary and Y is $(n+1)$-ary, then everyXY is an abbreviation for $\overline{\text{some}X\overline{Y}}$.

Using a context-free grammar, and compositional semantics, Purdy defines a fragment of English, each of whose sentences are interpreted by 0-ary $\mathcal{L}_N$ expressions. Sentences generated by this grammar can contain proper nouns, the determiners every, some and no, adjectives, relative clauses, (in)transitive verbs, passives, sentence and verb phrase coordination and phrases of the form there be VP, where VP is a verb phrase.

Under the assumption that the phrase is faster than expresses a transitive relation, the following argument is clearly valid. If we further assume that is faster than can be treated as a transitive verb, then we can express the argument in Purdy's fragment of English. A proof of its validity using $\mathcal{L}_N$ is given below, along with English glosses for each step.

Some horses are faster than some dogs
All dogs are faster than some men
Some horses are faster than some men

1	some$H$someDF	some horses are faster than some dogs
2	all$D$someMF	all dogs are faster than some men
3	$D \subseteq$someMF	all dogs are faster than some men
4	some$H$some(someMF)F	some horses are faster than some things faster than some men
5	some$H$(someMF)F	some horses are faster than some things faster than some men
6	$F \circ F \subseteq F$	for all pairs of things, the first being faster than something faster than the second implies the first being faster than the second
7	some$H$someMF	some horses are faster than some men

Purdy claims that every step of the above argument can be translated easily from English to $\mathcal{L}_N$ and back, and that, therefore, reasoning in $\mathcal{L}_N$ is a better model

for natural language reasoning than first-order logic. However, without details of how to perform the backwards translation, for some steps in the above argument (step 6, say), it is not clear that the English gloss given for the $\mathcal{L}_N$ formula is actually a member of Purdy's fragment of English. It is therefore difficult to see the difference between the production of this kind of gloss and the kind of *ad hoc* English glosses often attached to proofs in first-order logic.

We stated earlier that the formulae of Definition 2.1 only form a fragment of the full $\mathcal{L}_N$ – a fragment which can be embedded in Quine's fluted fragment of first-order logic. Roughly speaking, the fluted fragment is just first-order logic restricted so that variables occurring as arguments to non-unary predicates can occur *only* in the same order as the sequence of quantifiers binding them. So, for example, the formula

$$\forall x(p(x) \rightarrow \forall y(q(y) \rightarrow r(x, y)))$$

is fluted, whereas

$$\forall x(p(x) \rightarrow \exists y(q(y) \wedge \forall z(r(x) \rightarrow s(z, x, y))))$$

is not, since z appears before x and y as an argument to the ternary predicate s, but the quantifier binding z occurs within the scope of the quantifiers binding x and y. Fluted formulae are interpreted semantically in precisely the same way as general first-order formulae.

The translation into the fluted fragment of the $\mathcal{L}_N$ formulae in Definition 2.1 is straightforward. In particular, the insistence that variables occur in a fixed order in fluted formulae corresponds to the interpretation of n-ary $\mathcal{L}_N$ formulae relative to an *ordered* sequence of domain elements. It follows that, since satisfiability in the fluted fragment can be decided in non-deterministic exponential time, so can satisfiability in the fragment of $\mathcal{L}_N$ given above. In general, however, the full $\mathcal{L}_N$ contains "selection operators" – a means of permuting and selecting subsequences of the domain elements relative to which a $\mathcal{L}_N$ predicate is interpreted. These selection operators grant $\mathcal{L}_N$ the full expressive power of first-order logic with no restrictions on variable occurrence, and thus cause satisfiability to be undecidable.

McAllester and Givan, in [46], are concerned with identifying formal languages usable for automated reasoning which have both good computational properties and high expressive power. They suggest that such languages can be found by

seeking inspiration in the structures of natural language. In support of this claim, they define what they call *Montagovian syntax*, after Montague [84].

As with $\mathcal{L}_N$, the following definition does not give the full range of formulae allowed by McAllester and Givan, only those which they use to interpret English sentences. Full Montagovian syntax is expressively equivalent to first-order logic.

Definition 2.2. A *class expression* of Montagovian syntax is one of

1. a constant symbol
2. a unary predicate symbol
3. an expression of the form (R(some s)) or (R(every s)), where s is a class expression and R is a binary predicate symbol.

A *formula* of Montagovian syntax is one of

1. (every s w), where s, w are class expressions,
2. (some s w), where s, w are class expressions, or
3. Any Boolean combination of formulae of Montagovian syntax.

Expressions of Montagovian syntax are interpreted over structures containing a domain D of elements, in a similar fashion to formulae of first-order logic. Constants are interpreted as elements of D, unary predicates as subsets of D, and binary predicates as subsets of D^2. Complex class expressions can then be interpreted as subsets of D, with R (some s) denoting the elements of D which are in the relation R with some element in the denotation of s, and similarly for R (every s). Formulae can be assigned truth values in the obvious way: (every s w) is true if the denotation of s in D is a subset of the denotation of w, and false otherwise, and (some s w) is true if at least one element of D belongs to the denotations of both s and w, and false otherwise. Boolean combinations of formulae are assigned truth values via the usual interpretations of Boolean connectives. So

(every boy (loves (some girl)))

is true over a domain if every entity in the denotation of boy is in the loves relationship with at least one entity in the denotation of girl. Similarly, the formula

(every (loves (some girl)) (loves (some boy)))

is true precisely when the set of lovers of some girl is a subset of the set of lovers of some boy.

We can see from these examples that although some formulae in Montagovian syntax bear at least a passing resemblance to sentences of English, not all do – witness the second example.

McAllester and Givan go on to consider the satisfiability problem for certain sets of Montagovian formulae. In particular, they show that the satisfiability problem for formulae of the forms (every s w), (some s w) and their negations is NP-complete. That is to say, they consider the set of formulae restricted to those which do not contain conjunction and disjunction of other formulae. From a linguistic point of view, this restriction is equivalent to disallowing arbitrary Boolean combinations of sentences. The reason given for the NP-completeness of this problem is that it is not necessarily known in advance which class expressions might have an empty denotation in a satisfying structure. In the case where the set of formulae under consideration contains one of (some s s) or its negation for every class expression s, the satisfiability problem is decidable in deterministic polynomial time.

The relationship between the sets of formulae McAllester and Given consider, and sets of English sentences, is very loose. Although Montagovian formulae are often quite English-like, there is no very clear fit between them and any particularly natural fragment of English. Of course, such lack of fit is no real surprise: their intent is to study logic via natural language, not the other way around. Nonetheless, without a closer link to linguistic phenomena, it is hard to use Montagovian syntax to learn about the semantics of English.

A slightly curious claim is made in [46] regarding the relationship between choice of syntax and efficiency of inference. An advantage claimed for the use of Montagovian syntax over the corresponding fragment of first-order logic is that the former lends itself to more efficient inference. In support of this claim, the argument is made that certain inferences require, in first-order logic, higher-order unification, whereas the same inference in Montagovian formulae does not. For example, the Montagovian inference rule

$$\frac{\text{(every s t)}}{\text{(every (R (some s)) (R (some t)))}}$$

can be written in first-order logic as

$$\frac{\forall x(S(x) \rightarrow T(x))}{\forall y(\exists x(S(x) \wedge R(x,y)) \rightarrow (\exists x(T(x \wedge R(x,y)))))}$$

but, they claim, whereas Montagovian formulae such as

(every(child-of(some bird)) (friend-of (every bird-watcher)))

can be shown to be a premise for the first rule by ordinary unification, the corresponding first-order formula

$$\forall x(\exists y(\mathit{bird}(y) \wedge \mathit{child\text{-}of}(y,x)) \rightarrow (\forall y(\mathit{bird\text{-}watcher}(y) \rightarrow \mathit{friend\text{-}of}(y,x))))$$

can only be shown to be an instance of $\forall x(S(x) \rightarrow T(x))$ by unifying S with the higher-order expression $\lambda x.\exists y(\mathit{bird}(y) \wedge \mathit{child\text{-}of}(y,x))$ and Q with the higher-order expression $\lambda x.\forall y(\mathit{bird\text{-}watcher}(y) \rightarrow \mathit{friend\text{-}of}(y,x))$ and carrying out β-reduction. However, this observation does not reveal a general problem with first-order logic, but a problem with a particular choice of proof system for it: the same inference using, say, resolution applied to formulae in clause normal form, requires no higher-order unification.

As well as the preceding work, some attention has been paid to the problem of natural language reasoning using standard logical formalisms. In [6] and [5], Blackburn, Bos, Kohlhase and de Nivelle define a small fragment of English by means of a grammar implemented in the logic programming language, Prolog. This grammar simultaneously assigns first-order semantics to each sentence it generates, in a format which can be given as input to an automated theorem-prover. The motivation behind this approach is the desire to test semantic theories automatically. In particular, this system is used to experiment with theories of discourse semantics. For example, it is common to assume that a given discourse has an associated *context* – a body of background knowledge shared by participants, to which every new utterance in that discourse adds. Each such utterance is assumed to be *consistent* with the context, and *informative* – that is, it cannot be inferred from the existing context. The system proposed in [6] and [5] can be used to experiment with different ways to handle these constraints, and to fine-tune them to fit in with intuition. Modelling the context as a set of first-order formulae, the satisfaction of the consistency and informativity constraints on a fresh English sentence can be determined by translating it automatically into

first-order logic, and feeding both the result and the context into an automated theorem-prover, which is capable of determining consistency and deciding when the given formula can be inferred from the given context.

Blackburn, *et al*, do not study the precise logical fragment generated by their grammar, and so of course have no expectation as to whether the theorem-provers they use will terminate on any set of inputs. Nonetheless, the system they build provides a convenient method of experimenting with different semantic constraints on discourse without having to perform all of the reasoning manually, and thus it has the potential to assist in the understanding of the kinds of reasoning which actually do take place in natural languages.

The work on semantic complexity in this thesis continues the programme proposed by Pratt-Hartmann in [58], in which a small fragment of English based on the language of syllogistic reasoning is defined, and the complexity of its satisfiability problem, along with the complexity of several linguistically interesting extensions, is analysed. In particular, fragments including relative clauses, transitive verbs and anaphora, subject to various restrictions on anaphor interpretation, are shown to have semantic complexities ranging from deterministic polynomial time to undecidability. This work is continued in [60], with ditransitive verbs, and a similar approach is taken for English temporal expressions in [59]. Definitions of many of the fragments in [58] and [60], and statements of the corresponding results, are given in Chapter 5.

Notwithstanding these latter works, the general consensus seems to be that despite the common practice of using first-order logic to represent the semantics of natural language, it is not appropriate to use it to study the process of *reasoning*. The argument for this claim seems to rest on the view that humans reason using the *syntax* of their natural language. It is not obvious that such a claim is true. Indeed, it is not obvious that human reasoning is a syntactic process at all – it might, for example, operate directly on "internal" partial models. Rather than assuming untested psychological hypotheses regarding reasoning, we feel it is safer, yet just as instructive, to confine ourselves to a *descriptive* role. That is, provided that native speaker intuitions regarding truth conditions can be faithfully represented, the relation of entailment continues to hold between certain sets of English sentences regardless of the psychological process of checking that it does. This relation can be studied very effectively using the standard techniques of model theory. So we study natural language reasoning using a language

which makes the structure of potential models clear – that is, using first-order logic itself. A side-effect of this approach is that, given that first-order proof- and model-theory have received an enormous amount of attention, there is a large body of results of which we can take immediate advantage.

2.2 Expressive Power

Regarding questions of the expressive power of fragments of natural language, such work as there is appears to have been concerned largely with questions of quantification. In works such as [4], [39], [42], [85], and [88], expressive power has been characterised as follows. Given the assumption that all elements of a given syntactic category C of a given natural language denote mathematical objects of a certain kind, one can ask whether every object of that kind is in fact the denotation of some element of category C. If the answer is negative, then it is instructive to determine the limits of expressive power – that is, what are the conditions determining which of the full range of potential denotations are actually allowed in the language concerned. To take a specific example, if noun phrases or determiners of English are taken to denote quantifiers, it would be interesting to know whether there are mathematically-expressible quantifiers which cannot be expressed by noun phrases or determiners of English.

The classic reference for this question is of course Barwise and Cooper [4], where several constraints on quantifiers are identified, and a range of conjectures are made about their applicability to natural languages. The lines of enquiry begun there are continued in later works. In particular, Keenan and Stavi [39] and Keenan and Moss [42] examine a large class of English determiners and identify constraints which seem to apply to all of them. Conversely, van Benthem [88] examines various classes of mathematically-expressible quantifier and identifies which occur in English.

Let us now be more precise. Following van Benthem [88], but differing slightly from Barwise and Cooper [4], we take a *quantifier* Q over a domain E to be a *relation* between subsets of E. For example, if Q is all, then we can define

$$Q = \{(A, B) | A \subseteq E, B \subseteq E \text{ such that } A \subseteq B\}$$

and write $\mathsf{all}AB$ for $(A, B) \in \mathsf{all}$. We speak of A and B in this notation as being

the left and right arguments of Q, respectively. We also say that a quantifier A over a domain E can *express* a set of subsets $\mathcal{B}$ of E if for some $A \subseteq E$, QAB for all $B \in \mathcal{B}$.

Two specific properties of quantifiers seem to be very important in natural languages: *conservativity* and *monotonicity*.

Definition 2.3. A quantifier Q over a domain E is

1. *conservative* if for all $A, B \subseteq E$, $QAB = QA(B \cap A)$,
2. *right upward monotone* (MON↑) if for all $A, B, B' \subseteq E$, if QAB and $B \subseteq B' \subseteq E$, then QAB',
3. *right downward monotone* (MON↓) if for all $A, B, B' \subseteq E$, if QAB and $B' \subseteq B \subseteq E$, then QAB',
4. *left upward monotone* (↑MON) if for all $A, A', B \subseteq E$, if QAB and $A \subseteq A' \subseteq E$, then $QA'B$,
5. *left downward monotone* (↓MON) if for all $A, A', B \subseteq E$, if QAB and $A' \subseteq A \subseteq E$, then $QA'B$.

Conservativity (the "lives on" property of Barwise and Cooper) essentially limits the domain of a quantifier to its first argument. In interpreting all men are mortal, it is not necessary to consider any non-men; the truth conditions of this sentence depend solely on the set of men. The various forms of monotonicity describe how quantifiers behave with respect to sub- and supersets of their left and right arguments. For example, all is both MON↑ and ↓MON since for any A, B, if allAB, then allAB' for every set B' containing B, and all$A'B$ for every subset A' of A.. Similarly, some is both MON↑ and ↑MON since for any A, B, if someAB, then someAB' for every set B' containing B and some$A'B$ for every set A' containing A. Quantifiers such as these which have both a left and a right monotonicity property are said to be *doubly monotone*. Left upward and downward monotonicity are known by Barwise and Cooper as *persistence* and *anti-persistence*, respectively.

The property of conservativity appears to be fundamental to natural language quantification. It is hypothesised by Barwise and Cooper that every natural language determiner denotes a conservative quantifier. In [39], Keenan and Stavi

present a list of English determiners, which appears to be intended to be exhaustive, and show that, with the expected semantics, the closure of this list under Boolean combination and infinite conjunction and disjunction is precisely the set of all conservative quantifiers. In [42], Keenan and Moss extend the definition of conservativity to polyadic quantifiers, such as more than (as in more A than B are C), and show that all k-place English determiners denote conservative quantifiers. Of course, the truth of this claim does depend on the exhaustivity of the given lists of determiners, but so far, no counterexamples seem to have been found. Conservativity does not impose any real limits on expressive power, however: it is also shown in [42] that for any domain E, every set of subsets of E can be expressed by some conservative quantifier.

The properties of monotonicity appear to be nearly as widely applicable as conservativity. It is conjectured by Barwise and Cooper that all "simple" natural language determiners are right monotone, and, in general, this conjecture does seem to hold. However, there are relatively natural, though not necessarily simple, determiners such as an even number of which are easily shown to be neither right nor left monotone. Nonetheless, Väänänen and Westerståhl [85] show that an even number of can in fact be defined in terms of a Boolean combination of monotone quantifiers, even if those defining quantifiers do not themselves appear to be the denotations of natural language determiners.

Doubly monotone quantifiers are investigated in van Benthem [88], where it is shown that over finite domains, every doubly monotone quantifier is expressible as a Boolean combination of determiners of the form at most k out of every n A are B, from which it can be concluded that (at least) English can express all doubly monotone quantifiers. This result can be strengthened: it turns out that *every* quantifier over finite domains which can be defined in first-order logic can be expressed as a combination of determiners of the forms at most k (non-)A are (not) B and there are at most k (non-)A. This latter result is proved using the Ehrenfeucht-Fraïssé method to which we return in Chapter 3.

It is worth noting as an aside at this point that not all basic English determiners can be defined in first-order logic. The classic example is most. The quantifier denoted by most is right upward monotone and not left monotone at all, and is shown in Appendix C of Barwise and Cooper [4] not to be first-order definable.

Other classes of quantifier considered in [39] and [42] include *cardinal* and *logical* quantifiers. The set of cardinal quantifiers is taken to be those which

depend solely on the *number* of elements of their left argument which are in their right argument. For example, more than ten is a determiner denoting a cardinal quantifier. The set of logical quantifiers, which includes the cardinal quantifiers, is taken also to include those denoted by determiners such as every whose meaning does not depend crucially on the particular structure in which they are interpreted. By contrast, a possessive such as John's denotes a non-logical quantifier, since its meaning in a given structure depends on the interpretation of the symbol John in that structure. The following expressive power results are shown in [42]. Over infinite domains, the set of cardinal quantifiers is strictly smaller than the set of logical quantifiers, which is strictly smaller than the set of all conservative quantifiers. Over finite domains, however, cardinal and logical quantifiers have the same expressive power, although neither can express as much as the full set of conservative quantifiers.

The preceding discussion has hopefully given some flavour of the kind of results which can be obtained regarding the limitations of the expressive power of natural language when compared to the full set of mathematically-possible denotations. The results we present in this thesis are complementary to such work. An interesting direction for the future might be to see whether restrictions on quantification as discussed above could be used to generate restrictions, of the kind we consider here, on the possible models of English sentences.

2.3 Conclusion

Following [58], the work in this thesis is intended to form a systematic study of the inferential properties and expressive power of natural language. In contrast to the prevailing view that "logicians' logics" are not appropriate for the study of natural language reasoning, we conduct all of our work within the framework of first-order logic. No commitment to any claims regarding the psychological nature of human reasoning is made by this choice. Rather, it is simply that the underlying semantic structures can be perspicuously described in first-order logic in a way which we feel can be obscured by the use of formalisms more closely resembling English. Also, it is important that the techniques used to obtain our results be equally applicable to any natural language, and again, it seems as if a more English-like formalism would obstruct this goal.

We have, in this chapter, relied on a certain familiarity with notions such

as formula, structure, grammar, and so on. In the following two chapters, we present more formal accounts of the various mathematical and linguistic topics underlying our work. We begin with logic.

Chapter 3

Logic and Computation

...a very astute people...their ideas are ignoble, but they make no mistakes in carrying them out.

— John Butler Yeats

Let us begin by presenting the logical background needed for this thesis. In the following chapter, we go through a number of topics including: first-order logic and its semantics; various fragments of first-order logic, including modal logic; resolution theorem proving and some of its refinements; a discussion of ideas such as bisimulation and Ehrenfeucht-Fraïssé games used to analyse the expressive power of various logics; and some basic ideas of computational complexity .

3.1 First-order logic

3.1.1 First-order languages

The following completely standard definitions simply set out the basic terminology and notation to be used throughout.

Definition 3.1. A *signature* Σ is a set of relation/predicate symbols and function symbols, each of which has an associated positive *arity*. Function symbols of arity 0 are known as *constants*.

Given a signature Σ and a countable supply $X = x_1, x_2, \ldots$ of *variables*, the set of *terms* over Σ is defined recursively as follows. We simultaneously define the *depth* of a term.

Definition 3.2. 1. Every constant symbol is a term, of depth 1.

2. Every variable is a term, of depth 1.

3. If $n > 0$, f is a function symbol of arity n and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term, whose depth is the maximum of the depths of $t_1, \ldots, t_n$.

For any term t, denote the depth of t by $d(t)$.

The set of *formulae over* Σ is defined similarly. As with terms, let $d(\phi)$ be the depth of a formula ϕ.

Definition 3.3. 1. If p is a relation symbol of arity n and $t_1, \ldots, t_n$ are terms, then $p(t_1, \ldots, t_n)$ is a formula of depth $\max(d(t_1), \ldots, d(t_n))$.

2. If ϕ is a formula, then $\neg\phi$ is a formula of the same depth as ϕ.

3. If ϕ, ψ are formulae, then $\phi \wedge \psi$ and $\phi \vee \psi$ are formulae whose depth is the maximum of the depths of ϕ and ψ.

4. If ϕ is a formula and x_i is a variable, then $\forall x_i(\phi)$ is a formula of the same depth as ϕ. All variables in ϕ are said to be *in the scope* of $\forall x_i$.

The *first-order language over* Σ is the set of all formulae over Σ. We use the standard abbreviations: $\exists x_i(\phi)$ for $\neg(\forall x_i(\neg\phi))$, $\phi \to \psi$ for $(\neg\phi) \vee \psi$ and $\phi \leftrightarrow \psi$ for $(\phi \to \psi) \wedge (\psi \to \phi)$. Where the usual rules of precedence leave no ambiguity, we also omit unnecessary brackets.

An occurrence of a variable x_i in a formula ϕ is *bound* by the closest occurrence of $\forall x_i$ or $\exists x_i$ in whose scope it lies, and is simply *bound* if it is bound by some quantifier. A variable which is not bound in a formula is *free* in that formula. Usually the free variables are explicitly shown: let $x_1, \ldots, x_n$, be all the free variables of ϕ; we typically write $\phi(x_1, \ldots, x_n)$ instead of merely ϕ. The *universal closure* of $\phi(x_1, \ldots, x_n)$ is $\forall x_1 \ldots \forall x_n(\phi(x_1, \ldots, x_n))$ – the *existential closure* is defined similarly. A formula with no free variables is often called a *sentence* or a *closed formula*.

A formula containing no variables is *ground* and a formula containing at least one function symbol is *functional*.

Formulae matching rule 1 above – that is, formulae containing no quantifiers, disjunction, conjunction or negation – are called *atoms*. A *literal* is an atom or its negation. We write *unary literal* to mean a literal whose predicate is unary, and similarly for binary, ternary, and so on.

3.1.2 Semantics

We now describe how to assign semantics to a first-order language.

Definition 3.4. A *structure* $\mathfrak{A}$ interpreting a given signature Σ consists of a non-empty set A (the *domain* of $\mathfrak{A}$) and some fixed interpretation $s^{\mathfrak{A}}$ of every symbol s in Σ.

1. If c is a constant, then $c^{\mathfrak{A}} \in A$.
2. If f is a function symbol of arity n, then $f^{\mathfrak{A}}$ is a function $f^{\mathfrak{A}} : A^n \to A$.
3. If p is a relation symbol of arity n, then $p^{\mathfrak{A}}$ is a subset of A^n.

Definition 3.5. Let X be the set of variables and let $\mathfrak{A}$ be a structure. A *variable assignment* in A is a function $g : X \to A$. We abuse notation to apply a variable assignment g to an arbitrary term t in the following way.

1. If t is a variable x, then $g(t) = g(x)$.
2. If t is a constant c, then $g(t) = g(c^{\mathfrak{A}})$.
3. If $t = f(t_1, \ldots, t_n)$ for some n-ary function f and terms $t_1, \ldots, t_n$, then $g(t) = f^{\mathfrak{A}}(g(t_1), \ldots, g(t_n))$.

If g is a variable assignment, then for any variable x and any $a \in A$, let $g[x/a]$ be the variable assignment

$$g[x/a](y) = \begin{cases} g(y) & \text{if } x \neq y \\ a & \text{otherwise} \end{cases}$$

We are now in a position to give an interpretation of formulae in a structure.

Definition 3.6. Let ϕ be any formula over a signature Σ, let $\mathfrak{A}$ be a structure interpreting Σ and let g be a variable assignment in $\mathfrak{A}$. Then we say that ϕ *is true in* $\mathfrak{A}$, or ϕ *holds in* $\mathfrak{A}$, and write $\mathfrak{A} \models \phi[g]$ if one of the following (exhaustive) cases holds.

1. ϕ is of the form $p(t_1, \ldots, t_n)$ for some n-ary predicate p and terms $t_1, \ldots, t_n$ and $(g(t_1), \ldots, g(t_n)) \in p^{\mathfrak{A}}$.
2. ϕ is of the form $\neg\psi$ and it is not the case that $\mathfrak{A} \models \psi[g]$

3. ϕ is of the form $\psi_1 \wedge \psi_2$, and $\mathfrak{A} \models \psi_1[g]$ and $\mathfrak{A} \models \psi_2[g]$.

4. ϕ is of the form $\psi_1 \vee \psi_2$, and $\mathfrak{A} \models \psi_1[g]$ or $\mathfrak{A} \models \psi_2[g]$.

5. ϕ is of the form $\forall x(\psi)$ and for every $a \in A$, $\mathfrak{A} \models \psi[g[x/a]]$.

If ϕ is any formula such that for every variable assignment $g \in \mathfrak{A}$, $\mathfrak{A} \models \phi[g]$, then we say that ϕ is *true in* $\mathfrak{A}$, $\mathfrak{A}$ *models* ϕ or $\mathfrak{A}$ *is a model of* ϕ, and we write $\mathfrak{A} \models \phi$.

It is straightforward to show that for any closed formula ϕ, if $\mathfrak{A} \models \phi[g]$ for *any* assignment g in $\mathfrak{A}$, then $\mathfrak{A} \models \phi$. There is usually no need to refer explicitly to a variable assignment as we generally consider only closed formulae, or formulae with a small number of free variables. When $\phi(x_1, \ldots, x_n)$ is a formula with free variables $x_1, \ldots, x_n$, we tend to write $\mathfrak{A} \models \phi[a_1, \ldots, a_n]$ instead of $\mathfrak{A} \models \phi[g]$ for some variable assignment which maps x_i to a_i for all $1 \leq i \leq n$. It is easy to see from the above definition that it does not matter to which elements of A g maps the elements of $X \setminus \{x_1, \ldots, x_n\}$

Definition 3.7. Given a structure $\mathfrak{A}$ interpreting a signature Σ, let the *first-order theory of* $\mathfrak{A}$ be the set of all formulae over Σ true in $\mathfrak{A}$.

Definition 3.8. Let $\mathfrak{A}, \mathfrak{B}$ be structures such that $\mathfrak{A}$ interprets a signature Σ and $\mathfrak{B}$ interprets at least the symbols in Σ. If $A \subseteq B$, then $\mathfrak{B}$ is an *extension of* $\mathfrak{A}$, or $\mathfrak{A}$ is a *substructure of* $\mathfrak{B}$, if

1. for every constant c in Σ, $c^{\mathfrak{A}} = c^{\mathfrak{B}}$,

2. for every function f in Σ, $f^{\mathfrak{A}}$ is the restriction of $f^{\mathfrak{B}}$ to A.

3. for every relation r in Σ, $r^{\mathfrak{A}}$ is the restriction of $r^{\mathfrak{B}}$ to A.

We say that $\mathfrak{B}$ is an *elementary extension of* $\mathfrak{A}$ if $\mathfrak{B}$ is an extension of $\mathfrak{A}$ and for every first-order formula $\phi(x_1, \ldots, x_n)$ over Σ and sequence $a_1, \ldots, a_n \in A$, $\mathfrak{A} \models \phi[a_1, \ldots, a_n]$ if and only if $\mathfrak{B} \models \phi[a_1, \ldots, a_n]$.

Herbrand Structures

One method of actually building structures interpreting a signature Σ is from the elements of Σ themselves. Structures interpreting first-order languages in this way are commonly known as *Herbrand structures*.

Definition 3.9. Let Σ be a signature, which we assume always to contain at least one constant symbol c_0. The *Herbrand universe* H over Σ is constructed recursively.

1. For every constant c in Σ, $c \in H$.
2. If f is an n-ary function symbol in Σ and $t_1, \ldots, t_n \in H$, then $f(t_1, \ldots, t_n) \in H$.

Since $c_0 \in H$, H is non-empty and so can serve as the domain of a structure.

Definition 3.10. Let H be the Herbrand universe over a signature Σ. An *Herbrand structure* $\mathfrak{H}$ over Σ is a structure as in Definition 3.4 with domain H interpreting every term over Σ as itself and every n-ary relation symbol in Σ as an n-tuple of H.

Definitions 3.9 and 3.10 are essentially a way of getting semantic information from syntactic constructions, and allow us to show when purely syntactic processes have correctly generated the desired semantic result. For example, we use Herbrand structures later to extract models for formulae from the output of automated theorem proving techniques.

Entailment and Satisfiability

Definition 3.11. Let ϕ, ψ be closed formulae over a signature Σ and suppose that in every structure $\mathfrak{A}$ interpreting Σ, if $\mathfrak{A} \models \phi$, then $\mathfrak{A} \models \psi$. Then we say that ϕ *entails* ψ, or that ψ *follows from* ϕ, and write $\phi \models \psi$. By extension, if Φ is any set of closed formulae and ψ a closed formula, we write $\Phi \models \psi$ as shorthand for $\bigwedge \Phi \models \psi$.

Formulae ϕ and ψ are said to be *equivalent* if ϕ entails ψ and ψ entails ϕ – i.e., if both formulae are true in precisely the same structures.

If a formula ϕ is entailed by the empty set of formulae, then ϕ is true in *every* structure, in which case we say that it is a *theorem*, written $\models \phi$. If no structure $\mathfrak{A}$ exists such that $\mathfrak{A} \models \phi$, then ϕ is *unsatisfiable* or *inconsistent*. Otherwise ϕ is *satisfiable* (*consistent*).

The following fundamental result can be found in any good textbook discussing the model theory of first-order logic.

Theorem 3.12. ***Compactness*** *Let Φ be a set of first-order formulae. Then Φ is satisfiable if and only if every finite subset of Φ is satisfiable.*

In other words, an unsatisfiable set of formulae is always unsatisfiable for a finite reason.

Types and Saturation

Definition 3.13. Let Σ be a signature and let $\Phi(x_1, \ldots, x_n)$ be a set of formulae over Σ containing a sequence of free variables $x_1, \ldots, x_n$. We call $\Phi(x_1, \ldots, x_n)$ an *n-type* over Σ.

A structure $\mathfrak{A}$ interpreting Σ *realises an n-type* $\Phi(x_1, \ldots, x_n)$ if there exist $a_1, \ldots, a_n \in A$ such that $\mathfrak{A} \models \Phi[a_1, \ldots, a_n]$.

Loosely speaking an n-type realised by $a_1, \ldots, a_n$ in a structure $\mathfrak{A}$ consists of everything which can be said about $a_1, \ldots, a_n$ if Σ were expanded to include a set of constant symbols naming the $a_1, \ldots, a_n$ in $\mathfrak{A}$.

Of course, it is entirely possible for there to be a structure $\mathfrak{A}$ whose first-order theory is consistent with an n-type $\Phi(x_1, \ldots, x_n)$, but in which $\Phi(x_1, \ldots, x_n)$ is not realised. Some structures simply do not contain enough elements to realise all possible types.

Definition 3.14. Let $\mathfrak{A}$ be a structure interpreting a signature Σ, let $A' \subseteq A$ and let $\Sigma(A)$ be Σ augmented with a constant symbol a for each $a \in A'$. Let $\mathfrak{A}(A')$ be the extension of $\mathfrak{A}$ interpreting $\Sigma(A')$ in which every constant $a \in A'$ denotes itself. A *type $\Phi(x)$ over A'* is a 1-type over $\Sigma(A')$.

A structure $\mathfrak{A}$ is *ω-saturated* if, for every finite subset A' of A, every type over A' is realised in $\mathfrak{A}(A')$

Structures which are ω-saturated contain enough elements to realise all possible n-types. The following results are discussed in [52, Chapter 5].

Theorem 3.15. *Let $\mathfrak{A}$ be an ω-saturated structure. Then $\mathfrak{A}$ realises every n-type consistent with its first-order theory.*

It is also always possible to obtain an ω-saturated structure whenever one is needed.

Theorem 3.16. *Every structure $\mathfrak{A}$ has an elementary extension $\mathfrak{A}^*$ such that $\mathfrak{A}^*$ is ω-saturated.*

In various results in the main body of this thesis, we use the following slightly restricted notion of "type".

Definition 3.17. Let $n \in \mathbb{N}$ and let $\mathfrak{A}$ be a structure. If $a_1, \ldots, a_n \in A$, the *n-type of* $a_1, \ldots, a_n$ *in* $\mathfrak{A}$, denoted $\mathrm{tp}^{\mathfrak{A}}[a_1, \ldots, a_n]$, is the set of all quantifier-free formulae $\phi(x_1, \ldots, x_n)$ such that $\mathfrak{A} \models \phi[a_1, \ldots, a_n]$.

3.1.3 Fragments

It is often useful to restrict attention to subsets of the first-order language over a signature – for example, to obtain a language with good computational properties. We refer to such subsets as *fragments* of the full language. In the course of this thesis, as described in Chapter 1, we look at fragments of first-order logic generated by the translation of certain sets of English sentences, where the restrictions on logical formulae arise from the rules of English syntax. In this section, we describe several ways in which fragments of first-order logic have been defined directly by restrictions on the logical syntax.

Prefix fragments

One way in which first-order formulae can be restricted is in the number and sequence of quantifiers they contain. A fragment of first-order logic subject to such restrictions is known as a *prefix* fragment. Much of the early work on the decision problem for first-order logic (of which more later) involved the study of various prefix fragments, in the hope of shedding light on the general, unrestricted case. After the general decision problem was shown to be undecidable, this work continued in order to classify which prefix fragments are decidable, and which are not. A thorough overview of the results on prefix fragments, and of the decision problem in general, can be found in [8].

Definition 3.18. A first-order formula is said to be in *prenex normal form* if it is of the form $Q_1 x_1 Q_2 x_2 \ldots Q_n x_n(\psi(x_1, \ldots, x_n))$, where $Q_1, \ldots, Q_n$ are quantifiers, $x_1, \ldots, x_n$ are variables and $\psi(x_1, \ldots, x_n)$ is quantifier-free.

Theorem 3.19. *Every first-order formula is equivalent to a formula in prenex normal form.*

A prefix fragment of first-order logic is specified by giving a pattern of quantifiers, such as $\forall\exists\forall$ and (optionally) a restriction on the content of signatures, such

as "only unary predicates, no constants or function symbols". The fragment thus defined contains all and only the prenex normal formulae in signatures meeting the given restriction whose initial sequence of quantifiers match the given pattern. Taking the examples just given, the formula

$$\forall x \exists y \forall z (p(x) \rightarrow (q(y) \wedge r(z))),$$

where p, q, r are unary predicates, is a member of the specified fragment, whereas neither

$$\forall x \exists y \forall z (p(x) \rightarrow (q(y) \wedge r(z) \rightarrow s(x, z)))$$

nor

$$\forall w \exists x \forall y \exists z (p(w) \rightarrow (q(x) \wedge r(y) \rightarrow s(z)))$$

are, containing as they do non-unary predicates and too many quantifiers, respectively.

Limited-variable fragments

In Definition 3.3, we described languages whose formulae may contain an arbitrary number of variables, thus essentially allowing first-order formulae to quantify over an arbitrary number of sets. Limited-variable fragments result from attempts to constrain this expressive power.

Definition 3.20. Let $k \in \mathbb{N}$ and let Σ be a signature. The *k-variable fragment* over Σ, denoted $\mathcal{L}_k$, is the set of all formulae over Σ containing only the variables $x_1, \ldots, x_k$.

So, for example, the formula

$$\forall x_1 (man(x_1) \rightarrow mortal(x_1))$$

is in the fragment $\mathcal{L}_1$, whereas

$$\forall x_1 (man(x_1) \wedge \exists x_2 (donkey(x_2) \wedge own(x_1, x_2)) \rightarrow \forall x_2 (stick(x_2) \rightarrow \neg own(x_1, x_2)))$$

is in $\mathcal{L}_2$. Note how different occurrences of the variable x_2 are bound by different quantifiers in the second example.

Very roughly speaking, a limitation on variables sets a limit on the number of

entities between which a given formula can describe relations. As such, limited-variable fragments can be very useful for describing, for example, resource-limited computation.

As it turns out, many of the logical fragments we define in this thesis in terms of fragments of English turn out to be subfragments of $\mathcal{L}_2$. Specifically, all fragments not featuring ditransitive verbs fall into this category.

The basic modal fragment

The basic language of modal logic translates into a fragment of first-order logic, as do many of its extensions. Since some results on modal logic are of direct relevance to the current work, it is worth taking the time to give a reasonably full exposition. More details can be found in [7].

Definition 3.21. Given a set S of proposition symbols, the basic modal language over S contains all and only formulae generated by the following rules.

1. p is a modal formula for all $p \in S$.
2. If ϕ is a modal formula, then $\neg\phi$ is a modal formula.
3. If ϕ, ψ are modal formulae, $\phi \wedge \psi$ and $\phi \vee \psi$ are modal formulae.
4. If ϕ is a modal formula, $\Box\phi$ is a modal formula.

We take $\phi \rightarrow \psi$, $\phi \leftrightarrow \psi$ to be the same abbreviations as in Definition 3.3, and take $\Diamond\phi$ to abbreviate $\neg(\Box(\neg\phi))$. We omit unnecessary brackets where possible.

Definition 3.22. Let a modal structure $\mathfrak{M}$ consist of a set of *worlds* W, each of which is an assignment of *true (T)* or *false (F)* to each $p \in S$, and an *accessibility relation* $R \subseteq W \times W$. We say that a modal formula ϕ is *true at a world* $w \in W$ *in* $\mathfrak{M}$, written $\mathfrak{M}, w \models \phi$ if

1. $\phi = p$ for some $p \in S$ and w assigns T to p.
2. $\phi = \neg\psi$ for some modal formula ψ and it is not the case that $\mathfrak{M}, w \models \psi$.
3. $\phi = \psi_1 \wedge \psi_2$ for some modal formulae ψ_1, ψ_2 such that $\mathfrak{M}, w \models \psi_1$ and $\mathfrak{M}, w \models \psi_2$.
4. $\phi = \psi_1 \vee \psi_2$ for some modal formulae ψ_1, ψ_2 such that $\mathfrak{M}, w \models \psi_1$ or $\mathfrak{M}, w \models \psi_2$.

5. $\phi = \Box\psi$ for some modal formula ψ such that for every $w' \in W$ such that $(w, w') \in R$, $\mathfrak{M}, w' \models \psi$.

It follows that if $\phi = \Diamond\psi$ for some modal formula ψ, then $\mathfrak{M}, w \models \phi$ if there exists $w' \in W$ such that $(w, w') \in R$ and $\mathfrak{M}, w' \models \psi$.

Entailment, equivalence, satisfiability, and so on, are all defined similarly for modal formulae as for first-order formulae.

The *standard translation* transforms modal logic into a fragment of first-order logic, as we now show.

Definition 3.23. Let S be a set of proposition letters and let ϕ be a modal formula over S. The *standard translation* $\mathrm{ST}_x(\phi)$ is defined as follows. Let Σ be a signature constructed from S by taking every $p \in S$ to be a unary predicate in Σ, as well as a single binary relation r.

1. If $\phi = p$ for some $p \in S$, then $\mathrm{ST}_x(\phi) = p(x)$.
2. If $\phi = \neg\psi$, then $\mathrm{ST}_x(\phi) = \neg\mathrm{ST}_x(\psi)$.
3. If $\phi = \psi_1 \wedge \psi_2$, then $\mathrm{ST}_x(\phi) = \mathrm{ST}_x(\psi_1) \wedge \mathrm{ST}_x(\psi_2)$
4. If $\phi = \psi_1 \vee \psi_2$, then $\mathrm{ST}_x(\phi) = \mathrm{ST}_x(\psi_1) \vee \mathrm{ST}_x(\psi_2)$
5. If $\phi = \Box\psi$, then $\mathrm{ST}_x(\phi) = \forall y(r(x, y) \rightarrow \mathrm{ST}_y(\psi))$ for some fresh variable y.

Theorem 3.24. *Let ϕ be a modal formula. Then there exists a modal structure $\mathfrak{M}$ and a world w such that $\mathfrak{M}, w \models \phi$ if and only if there exists a structure $\mathfrak{A}$ and $a \in A$ such that $\mathfrak{A} \models ST_x(\phi)[a]$.*

We may therefore meaningfully refer to the fragment produced by the standard translation as the *modal fragment of first-order logic*.

3.2 Expressive Power

Formal languages are often defined first in purely syntactic terms, with a view to providing a semantic interpretation later. However, it is the semantics which are usually of most interest, and it would be useful in general to know the full range of semantic notions expressible in a given language. Likewise, it would also be helpful to be able to compare the expressive power of two languages, and hence as

a consequence, be able to tell when expressions of one language can be translated into expressions of the other while retaining the same semantics.

What we would like, then, is to be able to give a purely semantic characterisation of a language $\mathcal{L}$, such that we can tell precisely when expressions of any given other language are equivalent to an expression in $\mathcal{L}$. Even more usefully, perhaps, such a characterisation might allow us to determine precisely which semantic areas of interest a given language can be used to study.

One standard example of this kind of result is the characterisation of modal logic developed by Johan van Benthem in his PhD thesis, published as [86]. We give a brief overview of his work here.

Definition 3.25. Let $\mathfrak{M}, \mathfrak{N}$ be modal structures with worlds W, V and accessibility relations R, R', respectively, and let $B \subseteq W \times V$. We say that B is a *bisimulation* if the following hold for every $(w, v) \in B$.

1. for all $w' \in W$, if wRw', then there is a $v' \in V$ such that vRv' and $(w', v') \in B$,

2. for all $v' \in V$, if vRv', then there is a $w' \in W$ such that wRw' and $(w', v') \in B$,

3. for every proposition letter p, $\mathfrak{M}, w \models p$ if and only if $\mathfrak{N}, v \models p$

Write $\mathfrak{M}, w \sim \mathfrak{N}, v$ if $(w, v) \in B$ for some bisimulation B, in which case we say that w and v are *bisimilar*, and write $\mathfrak{M} \sim \mathfrak{N}$ if there is some bisimulation $B \subseteq (W, V)$.

Given the above definition, the following is an easy structural induction on formulae.

Lemma 3.26. *Let $\mathfrak{M}, w$ and $\mathfrak{N}, v$ be bisimilar. Then for every modal formula ϕ, $\mathfrak{M}, w \models \phi$ if and only if $\mathfrak{N}, v \models \phi$.*

The converse does not hold, but via a rather more involved argument, we get:

Lemma 3.27. *Let $\mathfrak{M}, w$ and $\mathfrak{N}, v$ be such that for every modal formula ϕ, $\mathfrak{M}, w \models \phi$ if and only if $\mathfrak{N}, v \models \phi$. Then $\mathfrak{M}, \mathfrak{N}$ have elementary extensions $\mathfrak{M}^*, \mathfrak{N}^*$, respectively, such that w^* is a world in $\mathfrak{M}^*$, v^* is a world in $\mathfrak{N}^*$ and $\mathfrak{M}^*, w^*$ and $\mathfrak{N}^*, v^*$ are bisimilar.*

Lemmas 3.26 and 3.27 can together be used to prove what is known as the *van Benthem Characterisation Theorem* ([7], [86], [90]):

Theorem 3.28. *A first-order formula ϕ is equivalent to (the standard translation of) a modal formula if and only if ϕ is invariant for bisimulation, where* invariant for bisimulation *means that any two bisimilar structures agree on the truth value of ϕ.*

With bisimulation, then, we have a relatively simple relation on modal structures which tells us exactly when two structures have the same modal theory. If we now wish to compare the expressive power of a different fragment of first-order logic with that of modal logic, we can do so by purely semantic means, without having to go via more awkward approaches, such as syntactic transformations between the languages.

The relation of bisimulation itself also sheds light on the properties of modal logic: in particular, its "local" nature. Modal truth really depends only on the kinds of world which are accessible from *one* given world.

So what we would like, for any language $\mathcal{L}$, is to be able to construct a relation $\sim_{\mathcal{L}}$ between structures playing the same role for $\mathcal{L}$ as bisimulation does for modal logic. One method for achieving this goal for a very wide range of languages is that of Ehrenfeucht-Fraïssé games ([14], [52], among others.) These allow us to build arbitrarily close finite approximations to the desired relation $\sim_{\mathcal{L}}$, from which generalisation to $\sim_{\mathcal{L}}$ itself is straightforward. We do not directly use this method in this thesis, but the basic idea provides a useful framework for thinking about this kind of problem, so we give a sketch here.

Let $\mathcal{L}$ be a language and suppose we have two structures $\mathfrak{A}$ and $\mathfrak{B}$ interpreting $\mathcal{L}$-formulae. We wish to know whether $\mathfrak{A}$ and $\mathfrak{B}$ have the same $\mathcal{L}$-theory – or more generally, what features any $\mathfrak{A}$ and $\mathfrak{B}$ must share in order to have the same $\mathcal{L}$-theory. To find out, consider a two-player *Ehrenfeucht-Fraïssé game* played on $\mathfrak{A}$ and $\mathfrak{B}$. The players are known as *Abelard* (the *spoiler*) and *Eloise* (the *duplicator*) (after a rather unfortunate pair of mediæval lovers, one of whom, Abelard, was a logician.) Abelard wishes to show that $\mathfrak{A}$ and $\mathfrak{B}$ are different with regard to $\mathcal{L}$, and Eloise that they are the same. A game lasts n rounds, each consisting of a move by each player, with Abelard always moving first. In each round, Abelard picks one of the two structures and names an element in its domain. Eloise must then find an element in the other structure satisfying some

condition, which in general depends on $\mathcal{L}$. If she can do so, she wins that round; if not, Abelard wins the game. If Eloise wins all n rounds, she wins the game.

In order to be useful, the condition which Eloise's choices should meet are designed to ensure that if she wins, the sequences of n choices made over the course of the game are indistinguishable by formulae of $\mathcal{L}$ up to some "complexity level". By complexity level, we mean measures such as the quantifier depth of a first-order formula, or the depth of nesting of modal operators $\Box$ and $\Diamond$ in a modal formula. If Abelard wins, there must exist some $\mathcal{L}$-formula of complexity level n assigned different truth values by $\mathfrak{A}$ and $\mathfrak{B}$.

Of course, the result of a single game tells us nothing: one of the players may simply have made poor choices. More interesting is the notion of *winning strategies*. Roughly, a player has a winning strategy for the n-round game if (s)he has a method for selecting moves which ensures a win regardless of the other player's choices. If Eloise has a winning strategy for the n-round game, she can always show that $\mathfrak{A}$ and $\mathfrak{B}$ agree on $\mathcal{L}$-formulae of complexity up to n. If she has a winning strategy for the infinitely-long game, then $\mathfrak{A}$ and $\mathfrak{B}$ must agree on all $\mathcal{L}$-formulae.

The preceding summary of Ehrenfeucht-Fraïssé games is deliberately vague: the precise rules of the game depend crucially on the language $\mathcal{L}$ under consideration. However, the general idea remains the same regardless of the language.

It is easy to see how bisimulations, then, characterise the winning strategies for Eloise for the appropriate game on modal structures, where $\mathcal{L}$ is the basic modal language: no matter what Abelard chooses, she can always choose a bisimilar element, which is guaranteed to have the right properties. Although we do not, of course, necessarily need to invoke these games in order to define and discuss bisimulation.

So it is with the results in this thesis. For each logical language $\mathcal{L}$ defined from a fragment of English whose expressive power we consider, we define a relation $\sim_{\mathcal{L}}$ between structures which defines the appropriate winning strategies for Eloise – that is, which corresponds precisely to the preservation of the truth of $\mathcal{L}$-formulae between structures. We then give a purely semantic definition of $\mathcal{L}$ by means of an Invariance Theorem for $\sim_{\mathcal{L}}$. The van Benthem characterisation theorem is the Invariance Theorem most relevant to modal logic.

3.3 Automated Theorem-Proving

The *entailment problem* for a logic is the question of whether the conjunction of a given set of formulae in that logic (the *premises*) entails another formula in that logic (the *conclusion.*) The *satisfiability problem* for a logic is the question of whether a given set of formulae in that logic has a model.

Automated theorem proving involves attempts to answer these questions in such a way that the answer can be implemented on a computer. For a full overview of this field, see [69]. We concentrate here only on variants of one technique – resolution theorem proving – for which [45] is a good source.

For logics equipped with negation, the entailment and satisfiability problems are dual: if Φ is a set of formulae and ϕ a formula, then $\Phi \models \phi$ if and only if $\Phi \cup \{\neg\phi\}$ is unsatisfiable.

Resolution-based methods of theorem proving operate on sets of formulae in a particular normal form (*clause normal form*) and decide whether the given set is satisfiable. By the above observation, the corresponding questions of entailment are thus also immediately answered. The popularity of resolution stems from its ease of implementation; the drawback is that resolution proofs are far from easy for humans to read.

3.3.1 Clause normal form

In this section, we rehearse the familiar notions of Skolemisation and clausal form.

Consider an arbitrary closed first-order formula ϕ_0. We transform ϕ_0 by stages into a set $\mathcal{C}$ of *clauses* such that ϕ_0 is satisfiable if and only if $\mathcal{C}$ is satisfiable. (So ϕ_0 and $\mathcal{C}$ are *equisatisfiable.*)

Firstly, using the (easily-checked) De Morgan equivalences

$$\neg(\phi \wedge \psi) \leftrightarrow (\neg\phi \vee \neg\psi)$$

and

$$\neg(\phi \vee \psi) \leftrightarrow (\neg\phi \wedge \neg\psi),$$

all negations occurring in ϕ_0 can be moved inwards to produce a formula ϕ_1 equivalent to ϕ_0 in which negation only occurs applied to atoms, and never to complex formulae.

By Theorem 3.19, we can convert ϕ_1 into an equivalent formula ϕ_2 in prenex

normal form. We now proceed to eliminate existential quantifiers from ϕ_2.

For every existentially-quantified variable x occurring in ϕ_2, let n be the number of universal quantifiers in whose scope x occurs, let $x_1, \ldots, x_n$ be the variables bound by those universal quantifiers and let s_x be a fresh function symbol of arity n. For each such x, replace every occurrence of x in ϕ_2 by the term $s_x(x_1, \ldots, x_n)$ and let ϕ_2' be the resulting formula.

Finally, let ϕ_3 be the result of removing all existential quantifiers from ϕ_2'. We call ϕ_3 a *Skolemisation* of ϕ_2, and each new function symbol introduced in the construction of ϕ_3, a *Skolem* function. The process itself, we call *Skolemisation*.

Theorem 3.29. *With the above construction, ϕ_2 is satisfiable if and only if ϕ_3 is satisfiable.*

Note that unless ϕ_2 contains no existential quantifiers, ϕ_3 is not *equivalent* to ϕ_2, since ϕ_2 may well hold in a structure which does not interpret any of the Skolem functions in ϕ_3, and so we cannot ask whether ϕ_3 is true in $\mathfrak{A}$. Equisatisfiability suffices, however.

Example 3.30. If $\phi_2 = \exists x(p(x) \wedge q(x))$, then the formula $p(c) \wedge q(c)$ where c is a constant, is a Skolemisation of ϕ_2.

Similarly, Skolemising $\phi_2 = \forall x \exists y(\mathit{boy}(x) \rightarrow (\mathit{girl}(y) \wedge \mathit{love}(x, y)))$ generates a formula of the form $\forall x(\mathit{boy}(x) \rightarrow (\mathit{girl}(f(x)) \wedge \mathit{love}(x, f(x))))$ where f is a unary function symbol.

Since every variable in ϕ_3 is universally quantified, and the relative order of universal quantifiers in such a formula makes no difference to its truth conditions, as a matter of notational convenience we omit all universal quantifiers, and write the largest quantifier-free subformula of ϕ_3 instead of ϕ_3 itself.

Finally, using the equivalence $((\phi_1 \wedge \phi_2) \vee \phi_3) \leftrightarrow ((\phi_1 \vee \phi_3) \wedge (\phi_2 \vee \phi_3))$, rearrange ϕ_3 into the form $\psi_1 \wedge \ldots \psi_n$, where each ψ_i, $1 \leq i \leq n$, is a disjunction of literals – i.e., put ϕ_3 into *conjunctive normal form*.

Let $\mathcal{C} = \{\psi_i | 1 \leq i \leq n\}$. Each element of $\mathcal{C}$ is known as a *clause*. We interpret logical notions such as entailment, equivalence, satisfiability, and so on, applied to sets of clauses by the corresponding notions applied to the universal closure of the conjunction of the elements of that set. By construction, then, $\mathcal{C}$ is equisatisfiable with our original formula ϕ_0.

Every clause is therefore of the form

$$\neg A_1 \vee \neg A_2 \vee \ldots \vee \neg A_i \vee A_{i+1} \vee \ldots \vee A_n$$

for some $n > 0$ and $1 \leq i \leq n$, where each A_j, $1 \leq j \leq n$ is an atom. A clause is *unit* if it contains precisely one literal, *positive* (respectively, *negative*) if it contains no negated atoms (respectively, only negated atoms) and *Horn* if it contains at most one non-negated atom. The terms *ground*, *functional*, and so on, apply to clauses just as they do to general formulae.

3.3.2 Unification

The unification algorithm is the core of resolution-based methods of theorem proving. It essentially provides a method for determining when a given expression (term, atom, literal) is more general than another given expression (of the same kind.)

Definition 3.31. A *substitution* θ is a mapping from the set of variables to the set of terms (over a contextually-specified signature.) We apply substitutions to terms as follows (using postfix notation.)

1. $c\theta = c$ for every constant c.
2. $x\theta = \theta(x)$ for every variable x.
3. If f is a function symbol of arity n and $t_1, \ldots, t_n$ are terms, then

$$f(t_1, \ldots, t_n)\theta = f(t_1\theta, \ldots, t_n\theta).$$

Similarly, we extend this notation to apply a substitution θ to atoms, literals and clauses as follows.

Definition 3.32.
1. If p is an n-ary relation symbol and $t_1, \ldots, t_n$ are terms, $p(t_1, \ldots, t_n)\theta = p(t_1\theta, \ldots, t_n\theta)$.
2. If A is an atom, then $(\neg A)\theta = \neg(A\theta)$
3. If $C = l_1 \vee \ldots \vee l_n$ is a clause, where each l_i, $1 \leq i \leq n$ is a literal, then $C\theta = l_1\theta \vee \ldots \vee l_n\theta$.

Definition 3.33. A term t_1 is an *instance* of a term t_2 if there is some substitution θ such that $t_1 = t_2\theta$, in which case we say that t_2 is *more general* than t_1.

A substitution θ is more general than a substitution θ' if there exists a substitution σ such that for any term t, $t\theta' = (t\theta)\sigma$.

We speak of instances, and comparative generality, of atoms, literals and clauses via the obvious extension to the above definition.

Example 3.34. Let x, y be variables, let c be a constant and let r be a binary relation symbol. The atom $r(x, y)$ is more general than $r(c, y)$ because if θ is any substitution mapping x to c and y to itself, $r(x, y)\theta = r(c, y)$. Atoms $r(x, y)$ and $r(y, x)$ are equally general, because the substitution θ' mapping x to y and y to x gives $r(x, y)\theta' = r(y, x)$ and $r(y, x)\theta' = r(x, y)$.

Definition 3.35. Let E_1, E_2 be a pair of terms, atoms or literals. We say that E_1 and E_2 *unify* if there exists a substitution θ such that $E_1\theta = E_2\theta$, in which case we say θ is a *unifier* of E_1 and E_2. If, for every unifier σ of E_1 and E_2, θ is more general than σ, then θ is a *most general unifier* (m.g.u.) of E_1 and E_2.

Theorem 3.36. *Let E_1, E_2 be any pair of terms, atoms or literals. Then either E_1 and E_2 do not unify, or we can construct an m.g.u. θ of E_1 and E_2.*

Example 3.37. The atoms $r(c, y)$ and $r(x, d)$, where c, d are constants and r is a binary relation, can be unified with m.g.u. θ mapping x to c, y to d, so that $r(c, y)\theta = r(x, d)\theta = r(c, d)$.

By definition, if E_1 and E_2 unify with m.g.u. θ, then $E_1\theta (= E_2\theta)$ is an instance of both E_1 and E_2.

3.3.3 Resolution

Bearing in mind that every variable occurring in a clause is implicitly universally quantified, the idea of comparative generality of expressions given above corresponds to intuition: the truth of a formula entails the truth of all less general formulae. Similarly, if a formula is unsatisfiable, then so is every formula more general than it. This observation forms the basis of the resolution method of theorem proving.

Definition 3.38. Let $\mathcal{C}$ be a set of clauses and let $C \vee l$, $C' \vee l' \in \mathcal{C}$, where C, C' are (possibly empty) clauses and l, l' are literals. The *resolution rule* is as follows

$$\frac{C \vee l \qquad\qquad C' \vee l'}{C\theta \vee C'\theta} \qquad \text{if } l\theta = \neg l'\theta \text{ for some m.g.u. } \theta$$

where l, l' are known as the *resolved-upon* literals and $C\theta \vee C'\theta$ is a *(binary) resolvent* of $C \vee l$ and $C' \vee l'$. If the resolvent of any two clauses is the empty set, we denote it by $\bot$ ("bottom") and treat it as equivalent to the set of clauses $\{p(c), \neg p(c)\}$ for some (any) unary predicate p and some (any) constant c.

Let $C \vee l \vee l'$ be a clause in the same notation as above. The *factoring rule* is as follows

$$\frac{C \vee l \vee l'}{C\theta \vee l\theta} \qquad \text{if } l\theta = \neg l'\theta \text{ for some m.g.u. } \theta$$

where $C\theta \vee l\theta$ is known as a *factorisation* of $C \vee l \vee l'$.

Resolution and factoring together form a *proof system* for first order logic without equality: in the above notation, $C \vee l$, $C' \vee l'$ and $C \vee l \vee l'$ are *premises*, and $C\theta \vee C'\theta$ and $C\theta \vee l\theta$ are the *conclusions*, of their respective rules. We speak of *deriving* conclusions from premises, and can construct *derivations* in the obvious way. It is hopefully clear from the earlier discussion that neither resolution and factoring derive conclusions which are more general than their premises.

Definition 3.39. Let $\mathcal{C}$ be any set of clauses. By (possibly infinite) iteration of the resolution and factoring rules, we can construct a set $R^*(\mathcal{C})$ containing every clause derivable from $\mathcal{C}$ by these rules, including the elements of $\mathcal{C}$ themselves. We call $R^*(\mathcal{C})$ a *saturated clause set* and describe its construction as *saturation under resolution and factoring*. For every clause $C \in R^*(\mathcal{C})$, we write $\mathcal{C} \vdash C$, read as "C is provable from $\mathcal{C}$".

Theorem 3.40. *Let $\mathcal{C}$ be a set of clauses. Then $\mathcal{C}$ is unsatisfiable if and only if $\mathcal{C} \vdash \bot$. We say therefore that resolution and factoring are both* sound *and* refutation-complete.

Example 3.41. Let c, d be constants, p, q unary predicates and r a binary relation. The set of clauses $\mathcal{C} = \{p(c) \vee p(c), \neg p(x) \vee q(x)\}$ is visibly satisfiable. We can compute $R^*(\mathcal{C})$ in two steps:

$$\dfrac{\dfrac{p(c) \vee p(c)}{p(c)} \qquad \neg p(x) \vee q(x)}{q(c)}$$

since any substitution unifies $p(c)$ with itself, the substitution $\theta : x \mapsto c$ unifies $p(c)$ and $p(x)$ and $q(c) = q(x)\theta$. Since the union $\mathcal{C}'$ of $\mathcal{C}$ with $\{p(c), q(c)\}$ is clearly closed under resolution and factoring, $R^*(\mathcal{C}) = \mathcal{C}'$.

Similarly, we can see that the set $\mathcal{D} = \{p(c), \neg p(x) \vee r(x,d), \neg r(c,d)\}$ is unsatisfiable. One proof of this claim using resolution takes the form

$$\dfrac{\dfrac{p(c) \qquad \neg p(x) \vee r(x,d)}{r(c,d)} \qquad \neg r(c,d)}{\bot}$$

These examples are of course extremely simple. The main reason for choosing them is that with more complex examples, proofs using resolution and factoring very quickly become large and unwieldy. A great many of the "human-friendly" structures of formulae are lost in Skolemisation and the conversion to clause normal form, and so the proofs generated by rules operating on clauses are typically much harder to read than, say, a natural deduction proof of the same result. The advantage, of course, is that resolution and factoring are both very simple rules to implement. They also lend themselves well to questions of decidability. That is to say, given a language $\mathcal{L}$, if resolution and factoring can only derive finitely many clauses from any set of $\mathcal{L}$-formulae, then we have an algorithm which can decide whether an arbitrary set of $\mathcal{L}$-formulae is satisfiable.

In the context of automation, smaller is naturally nearly always better, and so it would be helpful if resolution and factoring could be constrained in such a way as to generate smaller proofs in preference to larger ones, while retaining refutation-completeness. Various restrictions to the resolution rule are used to achieve this goal. For the purposes of this thesis, we consider *A-orderings* and *hyperresolution*. As with most of the material in this section, more detailed and comprehensive overviews can be found in [45] and [69].

Definition 3.42. Let $\preceq$ be a partial order on the set of atoms. We say that $\preceq$ is an *A-ordering* if

1. $\preceq$ is well-founded.
2. if $A_1 \preceq A_2$, then for every substitution θ, $A_1\theta \preceq A_2\theta$, in which case we say that $\preceq$ is *liftable*.

A-orderings can be extended to literals by ignoring any negation symbols.

Given an A-ordering $\preceq$, we define a refinement of resolution called $\preceq$-ordered resolution.

Definition 3.43. Let $\preceq$ be an A-ordering, let $\mathcal{C}$ be a set of clauses and let $C \vee l$, $C' \vee l' \in \mathcal{C}$, where C, C' are (possibly empty) clauses and l, l' are literals. The *$\preceq$-ordered resolution rule* is as follows

$$\frac{C \vee l \qquad C' \vee l'}{C\theta \vee C'\theta} \qquad \begin{array}{c}\text{if } l\theta = \neg l'\theta \text{ for some m.g.u. } \theta \\ l, \neg l' \preceq\text{-maximal in their respective clauses}\end{array}$$

Likewise, if $C \vee l \vee l'$ is a clause in the same notation, the $\preceq$-*ordered factoring rule* is as follows

$$\frac{C \vee l \vee l'}{C\theta \vee l\theta} \qquad \begin{array}{c}\text{if } l\theta = \neg l'\theta \text{ for some m.g.u. } \theta \\ l, l' \text{ both } \preceq\text{-maximal in } C \vee l \vee l'\end{array}$$

where $C\theta \vee l\theta$ is known as a *factorisation* of $C \vee l \vee l'$.

Appropriate selection of an A-ordering can reduce the search space for unsatisfiability proofs of a set $\mathcal{C}$ of clauses, and so, by the following result, perhaps allow satisfiability to be decided for fragments of logic on which unordered resolution does not terminate.

Theorem 3.44. *For any A-ordering $\preceq$, $\preceq$-ordered resolution and factoring are sound and refutation-complete.*

Another refinement of resolution which can reduce the search space compared to ordinary resolution is *hyperresolution*, which essentially works by combining several resolution steps into one larger step. Hyperresolution also has the benefit of doubling up as a technique for building particularly useful models for certain satisfiable clause sets.

Definition 3.45. Let $C_1, \ldots, C_n$ be a set of positive unit clauses and let D be a clause of the form $\neg A_1 \vee \ldots \neg A_n \vee D_1$, where for all $1 \leq i \leq n$, A_i is an atom and D_1 is a (possibly empty) non-negative clause. Suppose further that θ is an m.g.u. of C_i and A_i for all $1 \leq i \leq n$. Then $D_1\theta$ is a *hyperresolvent* of $C_1, \ldots, C_n$ and D.

$$\frac{C_1 \ldots C_n \qquad \neg A_1 \vee \ldots \vee \neg A_n \vee D_1}{D_1\theta}$$

If D_1 is empty, i.e., if $D = \neg A_1 \vee \ldots \vee A_n$, then the hyperresolvent of $C_1, \ldots, C_n$ and D is $\bot$.

Writing the premises of the hyperresolution rule as implications, we can see that hyperresolution is a form of high-powered *modus ponens*:

$$\frac{C_1\theta \ldots C_n\theta \qquad C_1\theta \wedge \ldots \wedge C_n\theta \rightarrow D_1\theta}{D_1\theta}$$

Theorem 3.46. *Hyperresolution is sound and refutation-complete.*

We mentioned earlier that hyperresolution can be used to construct models for certain satisfiable clause sets. Consider a satisfiable set of Horn clauses $\mathcal{C}$. Let $R^*_H(\mathcal{C})$ be the result of saturating $\mathcal{C}$ under hyperresolution, using the obvious analogue of Definition 3.39, and let $\mathcal{D}$ be the set of all ground instances of positive unit clauses in $R^*_H(\mathcal{C})$. We can view $\mathcal{D}$ as a representation of a structure $\mathfrak{H}$ over the Herbrand universe of $\mathcal{C}$ by letting a tuple $t_1, \ldots, t_n$ of terms belong to the denotation of an n-ary relation r if and only if $r(t_1, \ldots, t_n) \in \mathcal{D}$. It turns out that $\mathfrak{H}$ is a model of $\mathcal{C}$, and has the following useful property.

Theorem 3.47. *Let $\mathcal{C}$, $\mathcal{D}$ and $\mathfrak{H}$ be as above. Then $\mathfrak{H}$ is a* unique minimal model *of $\mathcal{C}$, in the sense that if $\mathfrak{A}$ is any model of $\mathcal{C}$, and $\mathcal{D}_\mathfrak{A}$ is the set of all ground atoms true in $\mathfrak{A}$, then $\mathcal{D} \subseteq \mathcal{D}_\mathfrak{A}$.*

Thus if saturation of a clause set $\mathcal{C}$ under hyperresolution tells us that $\mathcal{C}$ is satisfiable, then we can immediately extract a model of $\mathcal{C}$ from the results of saturation. In fact, even if $\mathcal{C}$ is unsatisfiable, it is an easy corollary of the above that $\mathfrak{H}$ is a model of the non-negative subset of $\mathcal{C}$ (which, of course, is always satisfiable – clearly, hyperresolution can only derive $\bot$ from a clause set containing at least one purely negative clause.)

3.4 Computational Complexity

We give here a very brief overview of some of the main ideas in the theory of computational complexity. For full details and discussion, see, for example, [21] and [50] .

Roughly, complexity theory concerns itself with questions of the worst-case performance of different algorithms as applied to varying sizes of input. Clearly, in order to formalise such questions, it is necessary to have a formal model of algorithms and their execution. We proceed with what is probably the most familiar such model. The following section mainly follows [50].

3.4.1 Turing Machines

We can picture a *Turing machine* as a length of tape beginning with a start ("left-most") square, and consisting of countably many squares thereafter, a read/write

head which has a state and which can move left or right on the tape, and a program which, given a current state and a symbol on the tape, tells the head its new state, which direction to move and which symbol to write on the tape.

Definition 3.48. A *deterministic Turing machine* $M = (K, \Sigma, \delta, k)$ consists of a finite set of *states* K, a finite *alphabet* Σ, a *transition function* δ from $K \times \Sigma$ to $(K \cup \{halt, yes, no\}) \times \Sigma \times \{left, right, wait\}$ such that $K \cap \Sigma = \emptyset$ and $\{halt, yes, no, left, right, wait\} \cap (K \cup \Sigma) = \emptyset$, and an *initial state* $k \in K$. We assume Σ always contains distinguished symbols *blank* and *start*.

An *input* to a Turing machine $M = (K, \Sigma, \delta, k)$ is a finite sequence of symbols from Σ, representing the contents of the tape before computation begins, with *start* written on the leftmost square of the tape and the head positioned above that square.

A *configuration* (l, Λ, P, n) of a machine M consists of a *current state* $l \in K$, a *left string* Λ of symbols in Σ representing the contents of the tape from the leftmost square up to and including the square beneath the head, a *right string* P of symbols in Σ consisting of all symbols to the right of the head, and a *current step count* $n \in \mathbb{N}$. Where P is finite, assume that every square to the right of the last symbol in P contains *blank*. Every machine M with input P always has initial configuration $(k, start, P, 0)$.

Let $(l, \Lambda\lambda, \rho P, n)$ be a configuration of a machine M, where $\lambda, \rho \in \Sigma$, Λ, P are possibly empty strings over Σ and suppose that $\delta(l, \lambda) = (l', \lambda', D)$. If $D = right$, let $\Lambda' = \Lambda\lambda'\rho$ and let $P' = P$; if $D = left$, let $\Lambda' = \Lambda$ and let $P' = \lambda'\rho P$; and if $D = wait$, let $\Lambda' = \Lambda\lambda'$ and let $P' = \rho P$. Then we say that M with configuration $(l, \Lambda\lambda, \rho P, n)$ *yields the configuration* $(l', \Lambda', P', n + 1)$ *in one step.* We generalise to "yields in n steps" in the obvious way.

Essentially, we interpret the "yields" relation between configurations as the execution of the program δ on a given input. The machine starts in its initial configuration, with state k and the head positioned at *start* on the leftmost square of the tape. The next configuration yielded by the initial configuration is then determined according to δ as described above, and then so on until the machine enters one of the states *halt*, *yes* or *no*, at which point execution terminates.

If we consider a *problem* Π to be a question which can somehow be encoded as the input to some Turing machine M, then M *solves* Π if, when M is executed on the encoded input, it terminates with the correct answer to Π, in the following sense: if Π is a "yes/no" problem, the final state of M is *yes* if and only if the

answer to Π is "yes" and *no* if and only if the answer to Π is "no"; if Π is not a "yes/no" question, then M terminates with final state *halt* and the contents of the tape form a (suitably-encoded) correct answer to Π. *M does not solve* Π if it either terminates with an incorrect answer, or never terminates.

In practice, we generally take "a problem Π" to mean a whole class of particular questions which can be encoded as input to a Turing machine, and refer to specific questions as *instances of* Π. For example, a problem might be "String reversal", and an instance of that could be "what is the reverse of the string pirate?"

If Π contains only "yes/no" questions, then we say that it is a *decision problem*. All of the problems discussed in this thesis are decision problems. If some Turing machine M solves a decision problem Π, we say that Π is *decidable*, and that M *decides* Π. Otherwise, Π is *undecidable*.

Machines satisfying Definition 3.48 in every respect save that δ is allowed to be an arbitrary relation between its domain and codomain, rather than specifically a function, are known as *non-deterministic*. Taking *configuration*, *input* and *yield* to mean the same thing for a non-deterministic machine as they do for a deterministic machine, a given configuration and input to a non-deterministic machine may not yield a unique configuration. Thus the same input may produce different answers from such a machine depending on which choices of "next" configuration are made at each stage of computation. A non-deterministic machine M solves a problem Π if there is guaranteed to be at least one run of M with input Π which gives the correct answer.

It is well-known that some problems cannot be solved by any Turing machine, deterministic or otherwise. In the decidable cases, it is useful to have a means of comparing the efficiency of different machines in solving the same problem. During execution, machines can use two resources: space (the total number of symbols required on the tape) and time (the total number of steps required to terminate.) Often we are less interested in the particular function representing the resources used by a machine solving Π as we are in the general class of functions into which it falls: polynomial, exponential, and so on. The so-called "big O" notation formalises this idea.

Definition 3.49. Let $f : \mathbb{N} \to \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$ be functions. Then we say that $f(n) = O(g(n))$ if there are positive integers c, n_0 such that for all $n > n_0$, $f(n) \leq c.g(n)$.

Definition 3.50. Let Π be a problem and suppose there is a Turing machine M and functions $f, g : \mathbb{N} \to \mathbb{N}$ such that, given any instance of Π as input of length n, M terminates in $O(f(n))$ steps and uses $O(g(n))$ boxes on the tape. Then we say that M implements an algorithm for Π of *time complexity* $O(f(n))$ and *space complexity* $O(g(n))$

3.4.2 Complexity

In general, the space and time required to solve a given instance π of a problem Π depends on three factors: the existence of a Turing machine which solves Π, the number of symbols in the on-tape encoding of π and, of course, the particular instance π itself.

If we want to characterise the "difficulty" of solving a particular problem Π in terms of the resources needed by a Turing machine to solve Π, then we need to take these factors into account. Generally, some instances of a problem are easier to solve than others, and so in order to be as general as possible, we need to consider the worst case: what is the hardest it can possibly be to solve Π? In this way, we insulate our analyses from the properties of particular instances of any problem.

As well, if more than one Turing machine exists to solve a problem, then there can be differences in efficiency. After all, if a problem is solvable, then there certainly exist ways of solving it which involve many redundant steps of computation. The existence of inefficient solutions is no guide to the computational properties of a problem.

We therefore analyse the *computational complexity* of a problem Π as follows. Firstly, can Π be solved by any Turing machine, and if so, what are the worst-case resource requirements of such a machine, as a function of the size of instances of Π as encoded on the input tape? An answer to these questions provides an *upper bound* on complexity. Such an upper bound might not, of course, represent the most efficient solution of Π. A *lower bound* on complexity can be established by showing that every instance π' of a problem Π', with a known (or strongly suspected) minimum complexity, can be rewritten as a similarly-sized instance π of Π which has the same solution as π'. If Π can be shown to have identical upper and lower bounds, then we know that we cannot (or we are unlikely to) find a better method of solving Π. We say "strongly suspected minimum complexity", and "unlikely to find" because certain extremely well-known key questions

of complexity theory remain open: in particular, many of the suspected lower complexity bounds for certain families of problem have not yet been shown to be optimal. We return briefly to this issue below.

Definition 3.51. A *complexity class* is a set of problems which have some common upper bound on their complexities. Classes we encounter during this thesis include:

PTIME the set of problems solvable by a deterministic Turing machine in time bounded by a polynomial function of the size of the input.

NPTIME as PTIME, but allowing non-deterministic machines.

EXPTIME the set of problems solvable by a deterministic Turing machine in time bounded by an exponential function of the size of the input.

NEXPTIME as EXPTIME, but allowing non-deterministic machines.

PTIME and NPTIME are commonly abbreviated to P and NP, respectively. Since every deterministic Turing machine is also non-deterministic, we know that

$$P \subseteq NP \subseteq EXPTIME \subseteq NEXPTIME$$

Although it is known that P $\neq$ EXPTIME, it is a famously open problem whether the rest of these inclusions are all proper. It is suspected that they all are. Problems in P are often labelled *tractable*.

We now make precise the idea of the *reduction* of one problem to another, required to establish lower complexity bounds.

Definition 3.52. A decision problem Π is *reducible* to a decision problem Π' if every instance π of Π can be encoded in $PTIME$ as an instance π' of Π' such that π and π' have the same answer.

Thus if Π is reducible to Π', then Π' is *at least* as hard to solve as Π, apart from a polynomial difference in complexity resulting from the reduction itself.

Definition 3.53. Let C be a complexity class. A problem Π is *C-hard* if every problem in C is reducible to Π.

A problem Π is *C-complete* if it is in C and it is C-hard.

Actually, the definition of hardness for PTIME and its subsets requires a stricter definition of reducibility than that given above, since ideally we would like reductions to have insignificant complexity compared to the problem being reduced. However, we never need to consider the P-hardness or otherwise of any problems in this thesis, and so Definition 3.53 suffices.

So, if a particular problem is hard for a complexity class, then it is among the hardest problems in that class, in that every other problem in the class reduces to it. We can thus derive lower complexity bounds: if all that is known is that Π is solvable in EXPTIME, then it might be possible for there to be a PTIME algorithm solving Π. However, if Π can be shown to be EXPTIME-complete, then certainly no PTIME algorithm for Π exists. If there were, then any problem Π' in EXPTIME could be solved in PTIME simply by reducing Π' to an equisolvable instance π of Π in polynomial time (possible since Π is EXPTIME-complete), and then solving π, also in polynomial time. But since P $\neq$ EXPTIME, no such algorithm exists. Thus the notion of completeness fixes both upper and lower bounds on the complexity of a problem.

Example 3.54. Given a language $\mathcal{L}$ for which a notion of satisfiability has been defined, the *satisfiability problem for* $\mathcal{L}$ is the following question: given an arbitrary (set of) formula(e) ϕ of $\mathcal{L}$, decide whether ϕ is satisfiable.

Let $\mathcal{L}$ be any language of classical propositional logic. Then the satisfiability problem for $\mathcal{L}$ is NP-complete.

Not all problems have an algorithmic solution, as Turing famously showed. The most relevant example to this thesis is the following:

Example 3.55. Let $\mathcal{L}$ be the language of first-order logic. Then there exists no Turing machine which can decide the satisfiability problem for $\mathcal{L}$. We say therefore that first-order logic is *undecidable*.

There are, however, decidable fragments of first-order logic (see, for example, [8]):

Example 3.56. Let $\mathcal{L}_2$ be the two-variable fragment of first-order logic. The satisfiability problem for $\mathcal{L}_2$ is NEXPTIME-complete.

3.5 Conclusion

The preceding sections together provide the main logical and mathematical background needed to prove the results in this thesis. In the next chapter, we move on to linguistics, and a discussion of the ideas necessary to define the fragments of English with which we are concerned.

Chapter 4

Natural Language Syntax and Semantics

ROMULUS: Scotios sunt weeds. CAESAR: Be quiet, boy, and do not put yore nom in the acusative its not grammer.

— Nigel Molesworth, *Back in the Jug Agane*

In order to define formally the fragments of English we wish to study, we need two things. First, we need a method of generating sentences of English containing only our choice of syntactic features, and second, we need a method of assigning semantics to the sentences generated. Both of these should allow as clear a presentation as possible.

4.1 Syntactic Framework

Loosely speaking, we say a *generative grammar* is a formal description of a system which produces grammatical sentences of a given language. There are many well-known frameworks for specifying such grammars: examples include the variants of transformational grammar (e.g. [24]), head-driven phrase structure grammar (HPSG) ([53], [54]) and categorial grammar (e.g., [75]). Each of these attempts in some way to provide an explanatory account of sentence formation in natural languages, often also covering the interface between syntax and other linguistics processes, such as semantics or pragmatics. The differences between these frameworks tend to reflect different views of how language works – whether syntactic

processes involve transformation between multiple levels of representation of a sentence, or whether a language consists of complex formation rules working on a simple lexicon, or simple formation rules with a complex lexicon.

However, for this thesis, we are not concerned overmuch with the mechanics of sentence construction *per se*, nor the cognitive processes of parsing. Given that none of the constructions we consider are particularly rare or syntactically complex, we can be confident that any sensible framework for generative grammar can cope with them. The semantics we assign in each case are also relatively straightforward, and we state any particular simplifications or assumptions we make. We need a grammar only to generate the set of sentences meeting some naturally-expressed specification – "all sentences containing a transitive verb with proper nouns as its subject and object", say – and to assign to each sentence the correct semantics. By "correct" semantics, we mean truth conditions which the average native speaker of English would accept for the sentences concerned.

To this end, then, the fragments presented in this thesis are given using standard context-free phrase structure grammars with movement rules, such as those used in, for example, [24] and [25], among many others.

The idea of a context-free grammar is well known, and straightforward to deal with. Figure 4.1 shows an example of such a grammar and the structure of the sentences (phrases of category S) which can be generated by it. We return to the concept of movement later.

The following definition can be found in Hopcroft and Ullman [33].

Definition 4.1. A *context-free grammar* G is a tuple (N, T, R, S), where N is a finite set of *non-terminal symbols*, T is a finite set of *terminal symbols*, R is a set of *(phrase structure) rules* of the form $n \to \nu$, where $n \in N$ and ν is a string of symbols from $N \cup T$, and $S \in N$ is the *start symbol*.

We can generate *strings of category* S by recursively evaluating the rules of G as follows. If $n \in N$ is a non-terminal symbol, $S_1, \ldots, S_m \in N \cup T$ and $n \to S_1, \ldots, S_m \in R$ is a rule in G, then n evaluates to the concatenation of the evaluations of $S_1, \ldots, S_m$. If some $S_i \in N, 1 \leq i \leq m$, then evaluation of S_i proceeds recursively. Every terminal symbol evaluates to itself. A string of category S is then a result of evaluating S in G.

Generally, of course, we can represent a grammar $G = (N, T, R, S)$ simply by a list of the rules R, as in Figure 4.1, and a specified start symbol, which in the example given is S.

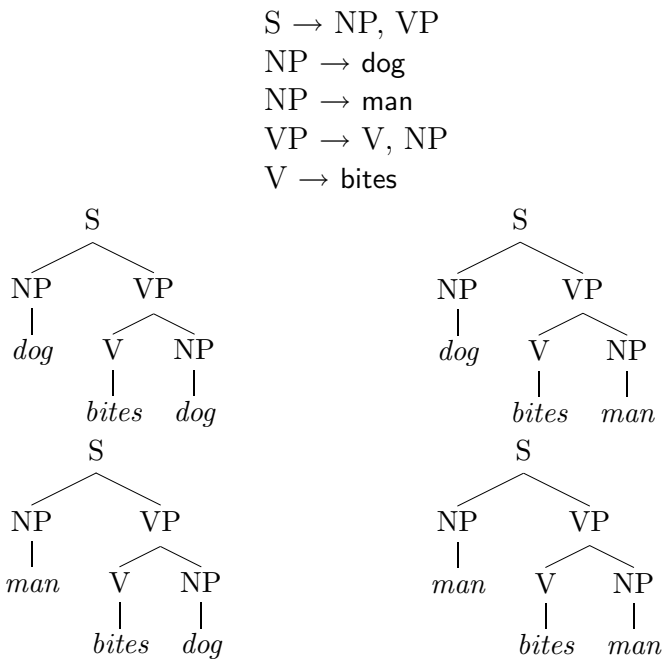


Figure 4.1: A sample grammar and the structures of all its generated sentences.

We divide every grammar into three sections. Rules with non-terminal symbols on the right-hand side form the *syntax*, and rules with only terminal categories on the right-hand side form the *lexicon*, which is made up of the *formal lexicon* and the *content lexicon*. The formal lexicon contains rules generating the closed class of grammaticalised words such as some, every, is, and not. The content lexicon contains rules generating the open classes of nouns and verbs.

When discussing the trees used to represent sentence structure, we need to refer to the structural property of *c-command*, which is defined as follows.

Definition 4.2. Let T be a tree and let A, B be nodes of T. We say that A *dominates* B if A is (strictly) higher in T than B, and A *c-commands* B if

1. A does not dominate B,
2. B does not dominate A,
3. the first branching node dominating A also dominates B.

For example, in Figure 4.2, NP_1 c-commands VP, V and NP_2, VP c-commands NP_1, V c-commands NP_2 and NP_2 c-commands V. The node S does not c-command any node.

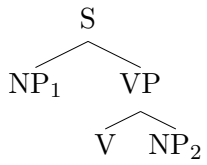


Figure 4.2: An example of c-command.

4.2 Semantics

Of course, as well as generating sentences, we also need to be able to assign a semantic representation to each sentence generated, preferably automatically. Using first-order logic as the representation language, the sentences generated by the grammar of Figure 4.1 ought to be assigned semantics according to Table 4.1.

dog bites dog	$bites(dog, dog)$
dog bites man	$bites(dog, man)$
man bites dog	$bites(man, dog)$
man bites man	$bites(man, man)$

Table 4.1: Semantics of a set of simple sentences

Such translation from sentences of natural language into logic is, of course, as old as logic itself, and a key component of any logic course involves a great deal of practice in translating one to the other in such a way that the translations have the same truth conditions as the original. We would like to handle this kind of translation both automatically and *compositionally*. That is, we would like the semantics of a phrase to be computed from the semantics of its constituents. In the example of Figure 4.1, we would like to assign semantics to dog, man and bites, and then be able to generate the semantics of VP and S from those assignments.

The tradition of compositional semantics is popularly held to go back to Frege, although the first fully fleshed out formalisation of the idea covering a wide range of kinds of sentences is in Montague [84]. Montague selected suitable expressions of higher-order logic to be the semantic interpretations of lexical items, and used higher-order composition to compute the semantics of a complex expression from the semantics of its parts. In particular, expressions which occur in a wide variety of linguistic contexts can be assigned semantics which give the desired result in any of those contexts.

Montague translated his fragments of English into *intensional logic*, a higher-order logic capable of expressing the intensional constructions those fragments included. Our concern is with entirely quantificational and relational features; we thus need only provide translations into first-order logic. We need certain higher-order constructs in order to do so, however.

Definition 4.3. Suppose we have a set of *types*, consisting of *basic types* e (the type of entities) and t (the type of truth values) and *functional types*, where a functional type is of the form $\langle \tau_1, \tau_2 \rangle$ for some basic or functional types τ_1, τ_2.

Let Σ be a first-order signature, and let a *typed variable* be a symbol not occurring in Σ which has an associated *type* τ.

The set of λ-expressions over a given signature Σ and set T of typed variables is defined recursively by the following conditions.

1. Every first-order term over the signature Σ and every element of T of type e is a λ-expression of type e.

2. Every closed first-order formula over the signature Σ and every element of T of type t is a λ-expression of type t.

3. If ϕ is a λ-expression of type τ_2, and x is a variable of type τ_1 in T possibly occurring in ϕ, then $\lambda x(\phi)$ is a λ-expression of type $\langle \tau_1, \tau_2 \rangle$. In this case we say that λx *binds* x in ϕ.

4. If ψ is a λ-expression of type τ_1, and ϕ is a λ-expression of type $\langle \tau_1, \tau_2 \rangle$, then $\phi(\psi)$ is a λ-expression of type τ_2. In this case we say that ϕ is *applied to* ψ.

Let ϕ, ψ be λ-expressions, let x be a variable of the same type as ψ, and let $\phi[\psi/x]$ denote the result of substituting ψ for every free occurrence of x in ϕ. More precisely, let $\phi[\psi/x]$ be

1. ψ if $\phi = x$,

2. ϕ if ϕ is of type e or t,

3. $\phi_1[\psi/x](\phi_2[\psi/x])$ if $\phi = \phi_1(\phi_2)$ for some λ-expressions ϕ_1, ϕ_2,

4. $\lambda x(\phi_1)$ if $\phi = \lambda x(\phi_1)$ for some λ-expression ϕ_1,

5. $\lambda y(\phi_1[\psi/x])$ if $\phi = \lambda y(\phi_1)$ for some λ-expression ϕ_1, $y \neq x$ and y not free in ψ or x not free in ϕ_1.

6. $\lambda z((\phi_1[z/y])[\psi/x])$ if $\phi = \lambda y(\phi_1)$ for some λ-expression ϕ_1, $y \neq x$, y not free in ψ, x not free in ϕ_1 and z a completely fresh variable of the same type as y.

Let ϕ be a λ-expression, and suppose that we can find an occurrence of $\lambda x(\phi_1)$ as a subexpression of ϕ. If y is a variable not occurring free in ϕ_1, of the same type as x, let ψ be the result of replacing the chosen occurrence of $\lambda x(\phi_1)$ in ϕ with $\lambda y(\phi_1[y/x])$. We call such a replacement a *change of bound variable in* ϕ. We say that λ-expressions ϕ and ψ of the same type are *α-equivalent* if ψ can be obtained by an empty or finite number of changes of bound variable in ϕ.

Let ψ be a λ-expression of type τ_1, and $\lambda x(\phi)$ a λ-expression of type $\langle \tau_1, \tau_2 \rangle$, so that x is a variable of type τ_1. We say that $\phi[\psi/x]$, which has type τ_2, is a *β-reduct* of $(\lambda x(\phi))(\psi)$ (which also has type τ_2), and call the process of computing β-reducts *β-reduction*. If ϕ, ψ are any λ-expressions of the same type, we say that ϕ *β-reduces* to ψ if ψ can be obtained from ϕ by an empty or finite sequence of β-reductions.

For example, if x has type e, c, d are constants and r is a binary predicate, then $\lambda x(r(c, x))$ has type $\langle e, t \rangle$ and d has type e. We should thus be able to apply $\lambda x(r(c, x))$ to d to obtain an expression of type t. The β-reduction of $\lambda x(r(c, x))[d]$ is $r(c, d)$ – a closed first-order formula, and hence, as required, of type t.

Similarly, let p have type $\langle e, t \rangle$, let x have type e and let $man, mortal$ be unary predicates. Then the λ-expressions $\lambda x(mortal(x))$ and $\lambda p(\forall y(man(y) \rightarrow p(y)))$ have types $\langle e, t \rangle$ and $\langle \langle e, t \langle, t \rangle$, respectively, as can be seen if we apply the latter to the former and compute the β-reduction. We begin with

$$\lambda p(\forall y(man(y) \rightarrow p(y)))[\lambda x(mortal(x))],$$

and replace every occurrence of p in $\forall y(man(y) \rightarrow p(y))$ with $\lambda x(mortal(x))$, to obtain

$$\forall y(man(y) \rightarrow (\lambda x(mortal(x)))(y)).$$

This expression contains another instance of λ-expression application, and so we

$\mathrm{S}/\phi(\psi) \rightarrow \mathrm{NP}/\phi,\ \mathrm{VP}/\psi$
$\mathrm{NP}/\lambda p(p(dog)) \rightarrow \mathsf{dog}$
$\mathrm{NP}/\lambda p(p(man)) \rightarrow \mathsf{man}$
$\mathrm{VP}/\phi(\psi) \rightarrow \mathrm{V}/\phi,\ \mathrm{NP}/\psi$
$\mathrm{V}/\lambda s \lambda x[s(\lambda y bite(x,y))] \rightarrow \mathsf{bites}$

Figure 4.3: A sample annotated grammar

β-reduce again, to get

$$\forall y(man(y) \rightarrow mortal(y)),$$

which has type t, as required.

Note that in order to avoid any possible clashes arising from two λ-expressions having variables in common, we assume in every instance of β-reduction that neither formula contains any typed variables occurring in the other. We silently replace any expressions violating this condition with an α-equivalent expression wherever necessary.

Expressions in the λ-calculus can be used to represent the semantics of words in natural language. The process of application can be used to compute the semantics of a phrase from the semantics of its components. For example, if Lara has the semantics $\lambda p(p(lara))$, and runs has the semantics $\lambda x(runs(x))$, then the semantics of Lara runs can be computed by applying the semantics of Lara to the semantics of runs, to produce $runs(lara)$.

We show the assignment of semantics to words and phrases by means of *semantically annotated grammars*. Suppose a grammar G contains a rule of the form $X \rightarrow X_1, X_2$, and a semantically annotated version G' of G contains a rule of the form $X/\psi(\phi) \rightarrow X_1/\phi, X_2/\psi$. Then G' is interpreted as saying that if X_1, X_2 are assigned semantics ϕ, ψ, then the semantics of X are computed by applying the semantics of X_2 to the semantics of X_1. The actual values of ϕ, ψ can be computed by recursively evaluating X_1, X_2 by further annotated rules of G'. If X_1, say, is a terminal category, then ϕ is a λ-expression. The generalisation to phrase structure rules with more than two symbols on the right-hand side is straightforward.

Figure 4.3 shows how the grammar of Figure 4.1 can be annotated with compositional semantics in this way, bearing in mind that for simplicity, we have assumed that both *man* and *dog* are proper nouns. To avoid clutter, we usually omit explicit reference to the types of λ-expressions.

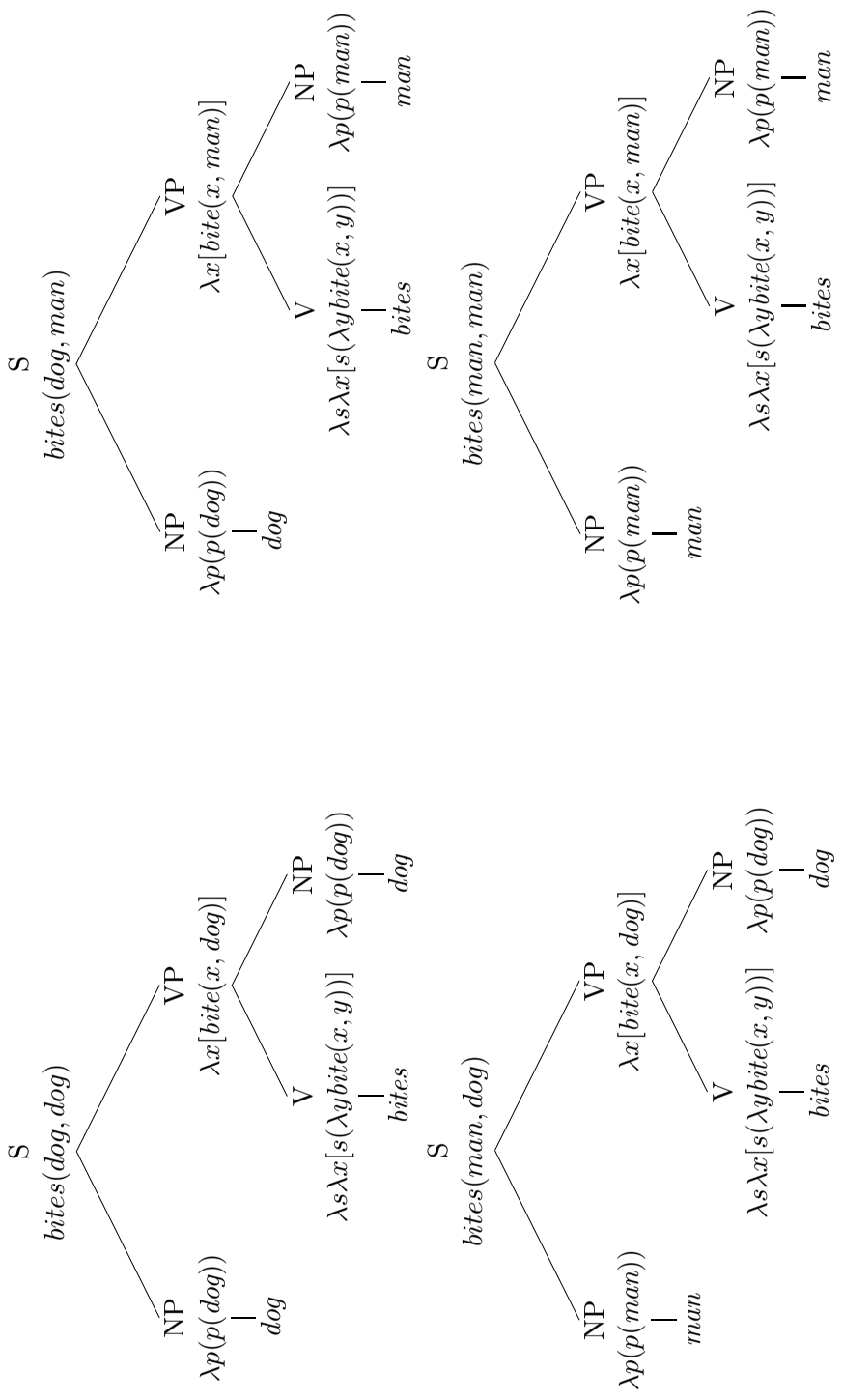


Figure 4.4: Computing semantics from an annotated grammar.

Let a *fragment* of natural language, or of logic, be a set of sentences, or of formulae, respectively. Figure 4.4 illustrates how a semantically annotated grammar generates a fragment of English, and a corresponding fragment of first-order logic. Since the first-order formulae generated are intended to represent the truth-conditions of the corresponding English sentences, we may use them to analyse the semantic properties of those sentences. For example, we can obtain a precise notion of entailment and equivalence between sentences by means of entailment and equivalence relations between formulae, and we can ask whether it is possible to tell if a given set of sentences is satisfiable. More precisely, for any fragment F of English, we transfer notions from first-order logic onto F as follows. Let Σ be any set of sentences of F and let Φ be the set of translations of Σ into first-order logic. Let $\|\Sigma\|$ be the number of symbols in Σ. We say that a sentence $S \in \Sigma$ is *true* (*false*) in a structure $\mathfrak{A}$ if and only if the logical translation $\phi \in \Phi$ of S is true (false) in $\mathfrak{A}$. The set Σ is *satisfiable* if and only if Φ is satisfiable, and we say that Σ *entails* a sentence S if Φ entails the semantics of S. For any structure $\mathfrak{A}$, let the *F-theory* of $\mathfrak{A}$ be the set of F-sentences true in $\mathfrak{A}$.

We can therefore speak of the *satisfiability problem* for a fragment of English in much the same way as we speak of it for a fragment of logic, and thus we can also inquire as to the computational complexity of solving such a problem, using the size $\|\Sigma\|$ of a set of sentences Σ as the complexity measure. For any fragment F of English, then, let the *semantic complexity* of F be the complexity of the satisfiability problem for F. Similarly, we can investigate the *expressive power* of a fragment of English, and attempt to prove Invariance Theorems of the kind described in Chapter 3, Section 3.2.

The benefit of using simple context-free grammars and Montague-style semantics is that both are widely known and relatively easy to follow. It is also straightforward to implement them as parsers in the logic programming language, Prolog, allowing quick and easy testing. We claim that these choices affect only the exposition: our results would have been identical had our choices been different.

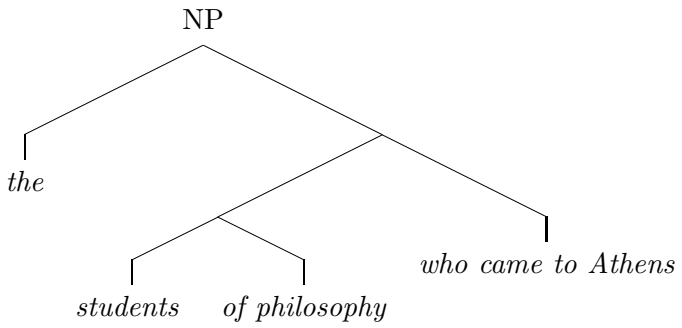
4.3 $\overline{X}$ Grammars

The grammars we present in this thesis are all generally within the standard $\overline{X}$ framework, a good description of which can be found in [24], among others. We give a brief introduction here.

The basic motivation is the observation that different phrasal categories – noun phrases, verb phrases, adjective phrases, and so on – all seem to have a common structure. Consider the sentence

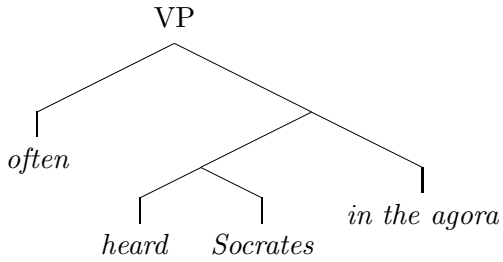
$[_S$ $[_{NP}$ The students of philosophy who came to Athens]
$[_{VP}$ often heard Socrates in the agora]]

and take the subject NP first. It consists of a common noun, students, modified or qualified by the, of philosophy and who came to Athens. Observing that of philosophy is intuitively more closely tied to the noun than who came to Athens – compare *the students who came to Athens of philosophy – and that the definite article is naturally taken to apply to the whole phrase, we propose the following structure.



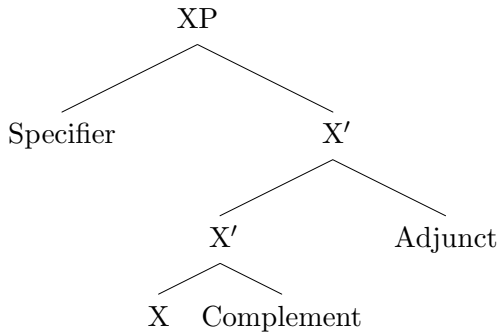
leaving the internal structures of the preposition phrase and the relative clause unexamined.

Now consider the verb phrase in the above sentence. It consists of a transitive verb heard, modified or qualified by often, Socrates and in the agora. As the direct object, Socrates has a closer relation to the verb than in the agora – compare *often heard in the agora Socrates – and the temporal adverb often is naturally taken to apply to the whole phrase. We therefore propose the following structure for the VP.



So here are two relatively complex phrases, of different types, both of which can plausibly be given the same structure. The conjecture behind the $\overline{X}$ framework is that this observation is not accidental, but reflects an underlying linguistic pattern, and that all phrases can be seen as having a similar structure.

So, let X be any basic syntactic category (leaving it deliberately vague what is meant by "basic") and let XP be a phrase whose main constituent is X (again, deliberately vague, but intended in the sense that a noun is the main constituent of a noun phrase, for example). We suppose that XP has the structure



Considering the common characteristics of the earlier examples, we take the *specifier* of X to be an element somehow restricting the whole XP, a *complement* of X to be an argument of, or element somehow necessarily connected to, X and an *adjunct* to be a modifier of X. X itself is known as the *head* of XP. A phrase need not necessarily contain all of these elements – Socrates, for example, is clearly a legal noun phrase on its own – but the claim is that phrases do not need to be given structures any more complicated than this one. Where necessary, we allow terminal nodes to contain "null" elements – elements which may have a semantic interpretation, and whose existence in the phrase structure is posited as having some explanatory power, but which do not correspond to a voiced component of the spoken sentence.

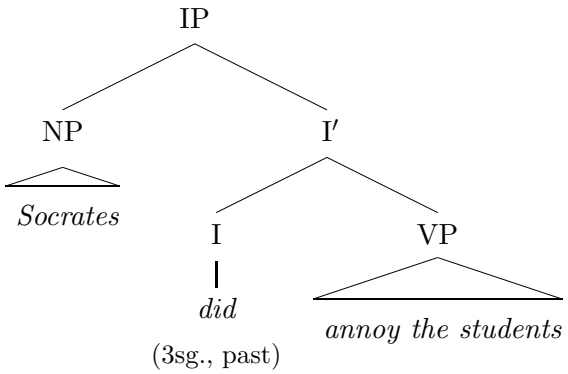
The X′ position (read "X bar" – the notations $\overline{X}$ and X′ are often used interchangeably) is proposed in order to capture the seemingly recursively defined

structure of many phrases. Adjuncts, it seems, can stack up – consider Figure 4.5.

Phrase structure rules generating such a structure fit the following pattern.

$$\begin{aligned} \text{XP} &\rightarrow \text{Spec, X}' \\ \text{X}' &\rightarrow \text{X}', \text{Adjunct} \\ \text{X}' &\rightarrow \text{X, Complement} \end{aligned}$$

The $\overline{X}$ structure extends to other syntactic categories easily. For example, the *sentence* is usually taken to be an *inflectional phrase (IP)*, headed by a tense/number inflection, or an auxiliary, whose specifier is the subject of the sentence and whose complement is the predicate. Sentences containing the auxiliary do provide a good example of this structure in English.



The grammars presented in this thesis are generally written within the $\overline{X}$-framework, primarily for reasons of clarity. No particular linguistic commitment is intended, however, and if, as occasionally happens, the clarity of a grammar can be improved by departing from the $\overline{X}$ structure, we do not hesitate to do so.

4.4 Specific Constructions

The constructions of English we need in order to define the fragments studied in this thesis are: proper and simply-quantified common noun phrases (Socrates, every man), the copula (is, is not), relative clauses (who is a stoic), transitive and ditransitive verbs(admire, prefer) and coordination (and, or). We handle all of these in a more or less standard way. Those cases which are slightly more involved, we explain here.

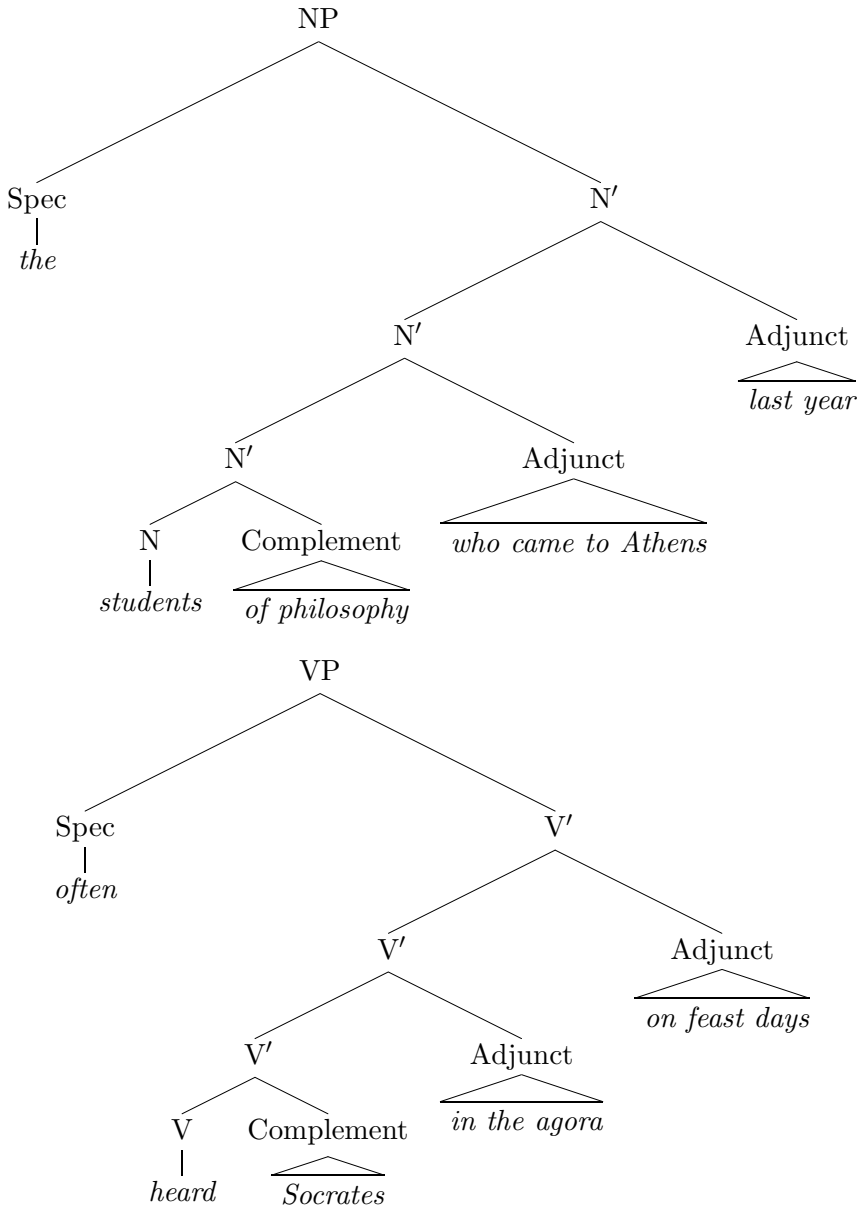


Figure 4.5: Recursive phrase structure

4.4.1 Relative Clauses

There are various situations in which constituents of a sentence do not seem to appear in their normal "base" position. For example, English specifiers normally precede their head, and yet with the analysis of a sentence as IP, with the inflection as head and the subject as specifier, simple yes/no questions seem to pose a problem.

Socrates did annoy the students
Did Socrates annoy the students?

Transformational accounts of syntax explain phrases such as these in terms of *movement*. Taking the structure of the first, declarative, sentence above as basic, they suggest that at some stage during syntactic processing, the auxiliary *moves* to a new position outside the IP. Other accounts of syntax – for example, categorial grammars – avoid the need to posit any kind of movement by dropping the idea that some sentence structures are more basic than others.

The handling of relative clauses is the only aspect of the work in this thesis which makes any use of movement rules. As we stated in the introduction to this chapter, it is unlikely that changing to a non-movement-based account would have any material effect on our results.

We take a *relative clause* to be a phrase of category IP – that is, a sentence. In order to allow an IP to occur as the *complement* to a common noun, we allow phrases of this category to be embedded within a *complementiser phrase*, *CP*, with a null head C. At least one main noun phrase – subject or (in)direct object – in the embedded IP must be one of the *wh-words* who, whom or which. The following phrase structure rules suffice.

$$\begin{aligned}
&\text{CP} \rightarrow \text{CSpec, C}' \\
&\text{CSpec} \rightarrow \\
&\text{C}' \rightarrow \text{C, IP} \\
&\text{C} \rightarrow
\end{aligned}$$

and each IP c-commanded by CSpec contains an NP generated by one of

$$\begin{aligned}
&\text{NP} \rightarrow \text{whom} \\
&\text{NP} \rightarrow \text{who} \\
&\text{NP} \rightarrow \text{which.}
\end{aligned}$$

Of course, sentences generated by such rules are not grammatical, in general. For example,

* Every man Socrates admires whom

In order to ensure correct English word order, we apply a rule of *wh-movement*. Thus, after generation, every wh-word occurring as an NP must be moved to the nearest c-commanding CSpec position, without passing through an intervening CP, and leaving behind a *trace* t, which is not pronounced and whose role is largely semantic. At most one wh-word can occupy any given CSpec position, and we insist that every CSpec position does contain a moved wh-word. Sentences whose parse trees cannot be made to satisfy these conditions are considered ungrammatical. We thus obtain the "deep" and "surface" structures for NPs, shown by example in Figure 4.6.

We often co-index the moved relative pronoun and the trace in examples for which the full tree structure is not shown, in order to indicate which movements have occurred. The noun phrase in Figure 4.6, for example, might be written as

$[_{NP}$ $[_{Det}$ every] $[_{N'}$ $[_{N}$ man] $[_{CP}$ $[_{CSpec}$ whom$_i$] $[_{C'}$ $[_{C}$] $[_{IP}$ Socrates admires t_i]]]]]

We also assume, without specifying a mechanism, that every transitive and ditransitive verb marks its NP complement(s) as *accusative* case and allow whom to occur only where it receives accusative marking. Admittedly, such case marking is very much a weakening aspect of modern English grammar, but we gain a small amount of extra clarity in our example sentences by using it here, and it costs nothing.

The conditions on wh-movement together ensure the (desired) ungrammaticality of examples such as

* every man Socrates admires whom
* every man who admires whom
* every man who whom admires

while still generating all desired cases.

Note also that we would like to rule out examples such as the following from being grammatical.

*$[_{NP}$ every man $[_{CP}$ who admires every sophist] $[_{CP}$ who despises Plato]]

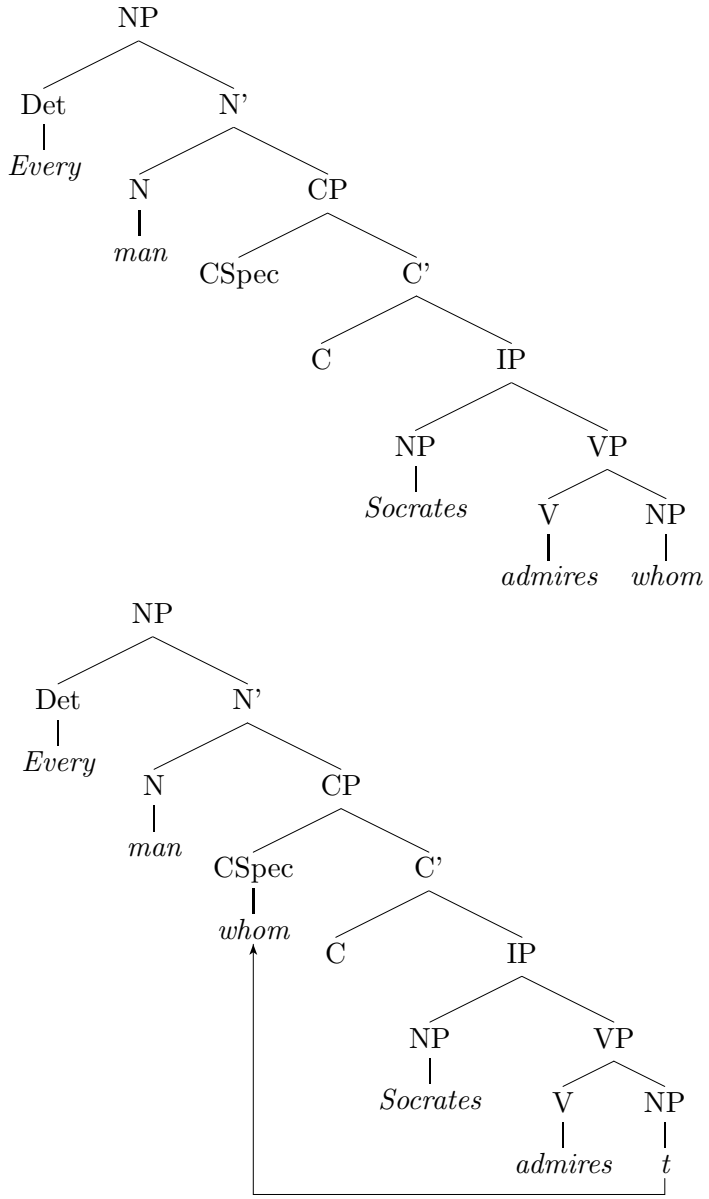


Figure 4.6: Sentence structure before and after wh-movement

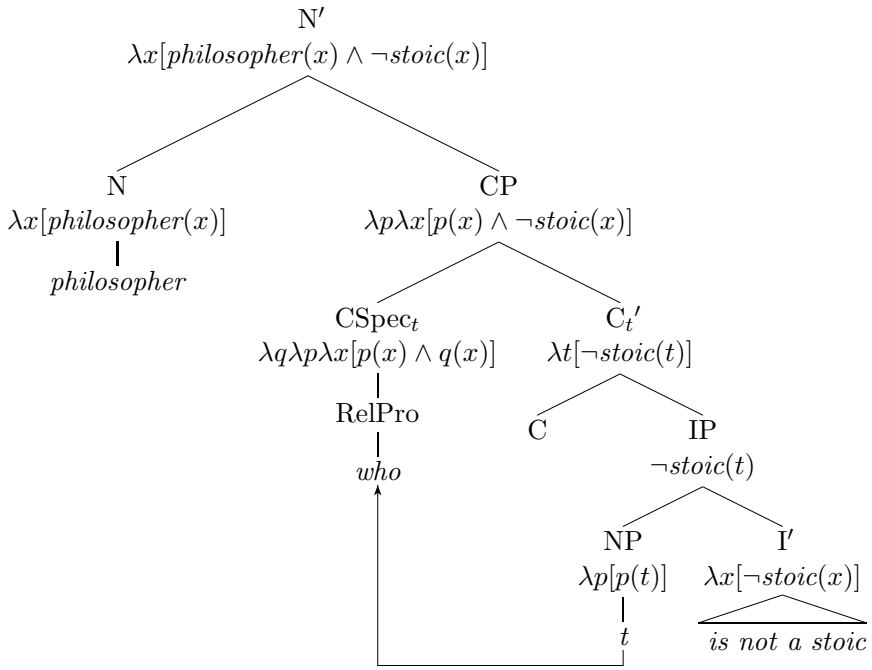


Figure 4.7: Computing the semantics of a relative clause (I)

That is, the reading in which who despises Plato qualifies man rather than sophist is not considered to be grammatical. We therefore design our grammars so that each noun can have at most one relative clause complement.

The handling of the semantics of wh-movement also requires some care. In order to maintain compositionality, the wh-trace must have a semantic representation of a type to which the semantics of a verb can be applied to yield the semantics of a VP. And yet, consider the desired semantics of the N′

$$\text{student whom}_i \text{ Socrates annoys } t_i$$

which ought to contribute a subformula of the form

$$\mathit{student}(x) \wedge \mathit{annoys}(\mathit{socrates}, x)$$

to the semantics of any sentence containing it. The second argument of *annoy* is the same variable which occurs as the only argument of *student*. We therefore need a way to mirror semantically the coindexation of trace and relative pronoun which occurs syntactically. To achieve this, we assume that each wh-trace t is assigned the semantics $\lambda p[p(t)]$, and that the semantics of the corresponding relative pronoun then bind exactly that term t to the desired value. Figures 4.7 and 4.8 show the computation of the semantics of phrases of category N′, and hopefully make it clear how such a process can work.

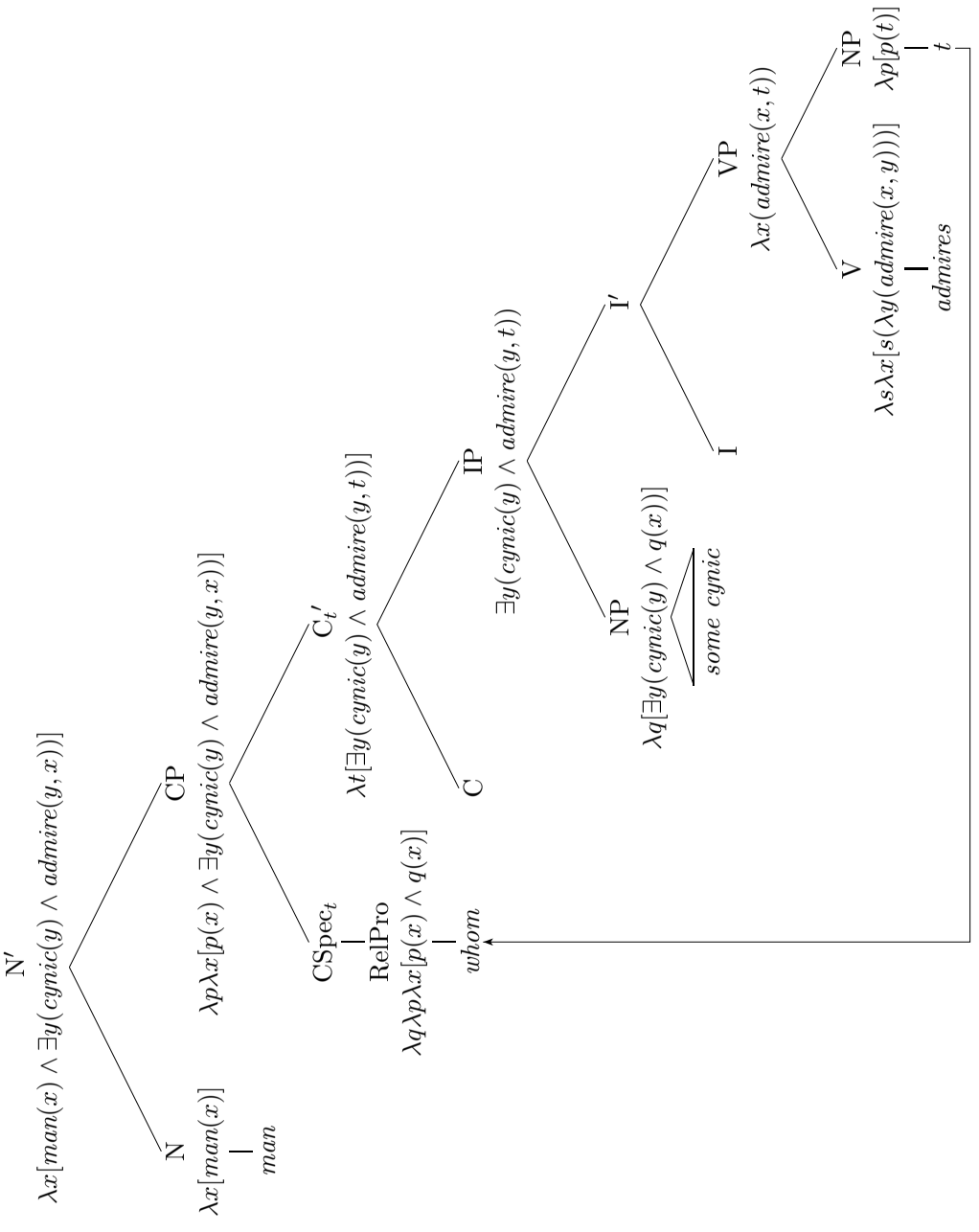


Figure 4.8: Computing the semantics of a relative clause (II)

4.4.2 Coordination

The main difficulty in handling coordination neatly is the breadth of its applicability. It seems that nearly any linguistic element can be coordinated with any other of the same category. For example, as well as straightforward coordination of sentences

Socrates is a mortal or I'll eat my hat

coordination can occur in a variety of subsentential positions.

Cataline was denounced by [$_{NP}$ Cicero] and [$_{NP}$ Tully]
Did you leave the country [$_{P}$ on] or [$_{P}$ before] the 1st of January 2000?
It was [$_{AdjP}$ extremely expensive] and [$_{PP}$ in bad taste]
Tell us, [are you now] or [have you ever been] a practising philosopher?

As the final examples illustrate, coordination is not limited to items of the same syntactic category, nor is it always between whole phrases. Data such as these make it particularly difficult to account for coordination using phrase structure grammars, although lexical systems such as categorial grammars can be more successful (for example, [93].) Within a transformational framework, [47] covers the main issues, and offers an account using movement and traces. However, in order to cope with all the data, the assumption that sentence structures are trees is abandoned, and more general graphs are used.

Within transformational theories, the application of transformations to coordinations is subject to a constraint identified in [70]. The Coordinate Structure Constraint demands that transformations apply equally to all coordinated structures.

For our purposes, we can largely avoid the difficulties, since we do not need a single, general-purpose coordination rule. In fact, such a rule might prove too broad for our needs. For example, semantically, of course, coordination of NPs must have a different effect to the coordination of verbs, and we wish to study the effects of each in isolation. Rather, then, than defining a generalised coordination rule and complicating our grammars in order to account for all the various difficulties which arise, we simply treat each kind of coordination separately. Thus, when we come to look at verb coordination, we need only the comparatively simple rules

$$\mathrm{V} \rightarrow \mathrm{V},\ \textsf{and},\ \mathrm{V}$$
$$\mathrm{V} \rightarrow \mathrm{V},\ \textsf{or},\ \mathrm{V}$$

along with the appropriate semantic annotations. Although these rules can be written in a way which satisfies the $\overline{X}$ structure, in the limited contexts we consider, it is hard to see what could be gained by doing so. We therefore prefer the simpler approach just described.

The only transformation we use is wh-movement. The only possible situations in which Ross' Coordinate Structure Constraint could apply, then, would be if we allowed coordination of relative pronouns, or of traces, or if a coordination intervened between the base position of a relative pronoun and its final landing site in CSpec. Coordination of silent elements such as traces is simply forbidden – * some man who$_i$ Socrates and t_i saw Plato – and coordination of wh-words makes sense only in contexts such as questions where the agreement features of the questioned element are not known, such as in Who$_i$ or what$_i$ did you see t_i that night?. In situations, such as those arising in Chapter 6, where an intervening coordination might affect wh-movement, we make appropriate modifications either to the relevant grammar or to the wh-movement transformation, to ensure that the Coordinate Structure Constraint is always satisfied.

4.5 Conclusion

Having briefly explored the relevant topics of both logic and linguistics, we are now in a position to formulate precise questions concerning the semantic complexity and expressive power of fragments of English. The following notation is used throughout the remainder of this thesis. For any fragment F of English, let an *F-sentence* be an element of F and let an *F-formula* be the result of translating an F-sentence into first-order logic. We also use F ambiguously to refer to the fragment of logic so generated, leaving context to indicate in each case which is meant.

We define a specific fragment F by fixing the syntax and formal lexicon; the content lexicon can vary. The choice of a particular content lexicon is equivalent to the selection of a signature for the corresponding first-order language. The label F therefore refers in fact to a *family* of fragments. When we speak of "the fragment F" in the absence of a contextually-salient content lexicon, we mean the union of all such fragments.

For brevity, we often assume that the content lexicon and the associated first-order signature always contain the same symbols, so that, for example, no distinction is drawn between the proper noun Socrates and the constant symbol *socrates*. This assumption is intended simply to avoid having to make unnecessary repeated reference to the translation process, and does not have any material consequences on the results presented.

In the next chapter, we define a range of fragments of English, featuring sentences whose main connectives are the copula, relative clauses and (di)transitive verbs, defining each fragment using a semantically annotated context-free grammar. For each fragment, we analyse the semantic complexity, and, by means of a notion of *simulation* between semantic structures, provide a characterisation of expressive power. These latter results can then be used to give completely semantic definitions of the fragments concerned.

Chapter 5

The Copula, Relative Clauses and Verbs

All syllogisms have three parts.
Therefore, this is not a syllogism.
— Unknown

In this chapter, we consider which fragments of English we can construct by taking simple NPs – proper nouns and simply-quantified common nouns – and forming sentences using the copula, relative clauses and both transitive and ditransitive verbs. For each fragment so constructed, we discuss its semantic complexity and characterise its expressive power using the techniques outlined in the preceding chapters.

We begin with an extremely simple fragment based on the copula, and proceed to extend that fragment, first with transitive, and then ditransitive, verbs, then with relative clauses and no verbs, before finally looking at the fragments we can generate using both relative clauses and verbs in combination.

Many of the semantic complexity results presented here were shown in [58] and [60], in which case we simply state the theorems without proof.

5.1 The syllogistic fragment: Cop

The earliest known fragment of a natural language to be studied for its logical properties is of course that fragment of ancient Greek used by Aristotle to define

the syllogistic form of reasoning. It therefore seems fitting that we begin with the corresponding fragment of modern English, which additionally has the advantages of being both syntactically and semantically simple. We are concerned initially, then, with sentences of the forms

c is (not) a p	Some p is (not) a q
Every p is a q	No p is a q

where p and q are common (count) nouns, and c is a proper noun. The fragment consisting of all such sentences was first introduced in [58], with the label $\mathcal{E}_0$. Using such sentences, we are able to express all possible syllogistic arguments, both valid and invalid:

Every man is a mortal	Every man is a mortal
Socrates is a man	Xanthippe is not a man
Socrates is a mortal	Xanthippe is not a mortal

The validity (respectively, invalidity) of these arguments corresponds to the logical impossibility (respectively, possibility) that all of the premises and the negation of the conclusion can hold simultaneously.

Every man is a mortal	Every man is a mortal
Socrates is a man	Xanthippe is not a man
Socrates is not a mortal	Xanthippe is a mortal

Each of the sentence-forms above admits an obvious translation into first-order logic, and by considering the possible formulae arising as such translations, we obtain a fragment of logic, which we can then use to prove results about the semantics of the original English fragment.

We begin with a formal definition of the language of the syllogism, and its semantics, via the following phrase structure rules

Syntax	Formal lexicon
$\mathrm{IP}/\phi(\psi) \rightarrow \mathrm{NP}/\phi,\ \mathrm{I}'/\psi$	$\mathrm{Det}/\lambda p\lambda q[\exists x(p(x) \wedge q(x))] \rightarrow$ some
$\mathrm{I}'/\phi \rightarrow$ is a , N'/ϕ	$\mathrm{Det}/\lambda p\lambda q[\forall x(p(x) \rightarrow q(x))] \rightarrow$ every
$\mathrm{I}'/\neg\phi \rightarrow$ is not a , N'/ϕ	$\mathrm{Det}/\lambda p\lambda q[\forall x(p(x) \rightarrow \neg q(x))] \rightarrow$ no
$\mathrm{NP}/\phi \rightarrow \mathrm{PropN}/\phi$	
$\mathrm{NP}/\phi(\psi) \rightarrow \mathrm{Det}/\phi,\ \mathrm{N}'/\psi$	
$\mathrm{N}'/\phi \rightarrow \mathrm{N}/\phi.$	

Content lexicon

$\mathrm{N}/\lambda x[man(x)] \rightarrow$ man	$\mathrm{PropN}/\lambda p[p(socrates)] \rightarrow$ Socrates
$\mathrm{N}/\lambda x[mortal(x)] \rightarrow$ mortal	$\mathrm{PropN}/\lambda p[p(diogenes)] \rightarrow$ Diogenes
$\vdots$	$\vdots$

The rules given here for I′ phrases deviate from the $\overline{X}$ framework, by treating is a and is not a essentially as single lexical items rather than complex phrases. The reason for handling the copula in this way is that we want to use it only in its predicative sense, and not in any of its other senses. For example, we do not use it to express identity, as in Cicero is Tully, or as an auxiliary, as in Cicero is going to the Curia. We also want to avoid the problems which might arise if we treated it as an ordinary transitive verb – for example, we would like to rule out sentences such as these:

* Every man is some mortal

* Socrates does not be a woman

* Every man is not Socrates

We therefore do not give separate rules for the copula, its negation, and the indefinite article, followed by a filter on generated sentences to eliminate output such as the above.

Figure 5.1 illustrates the derivation of a sentence (IP) in this fragment, along with the simultaneous generation of its semantics.

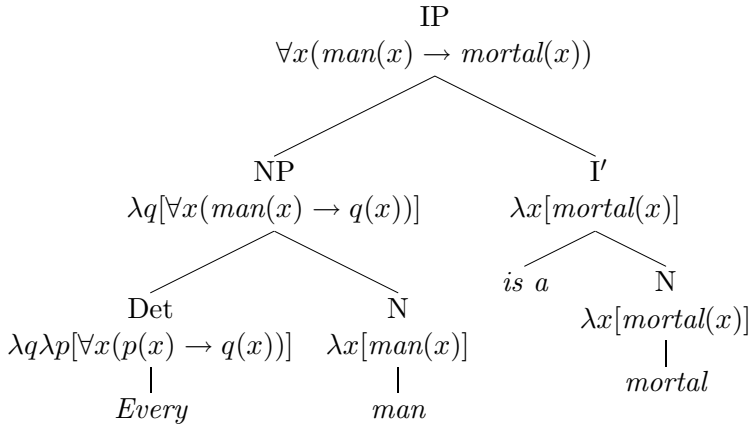


Figure 5.1: Structure of a simple Cop-sentence

It is straightforward to check that all the sentence-forms given earlier can be generated by such a process, and assigned the appropriate semantics, thus:

c is (not) a p	$\pm p(c)$
Some p is (not) a q	$\exists x(p(x) \wedge \pm q(x))$
Every p is a q	$\forall x(p(x) \rightarrow q(x))$
No p is a q	$\forall x(p(x) \rightarrow \neg q(x))$

In addition, the following slightly more awkward sentences are also generated.

Every p is not a q	$\forall x(p(x) \rightarrow \neg q(x))$
No p is not a q	$\forall x(p(x) \rightarrow q(x))$

However, as both of these sentence-forms are assigned the same semantics as less awkward forms listed above, this overgeneration need not concern us: no tightening of the grammar can have any effect on the logic.

We take all such semantic assignments to be uncontroversial. The occurrence of precisely the above translations as exercises in countless introductory logic courses provides sufficient evidence for their adequacy and general acceptance. In a fragment as simple as this one, it is particularly easy to see the relation between the structure of the English sentences and the structure of the logic: proper nouns give rise to constants, the determiners every and no to universal quantifiers and the determiner some to an existential quantifier.

The common grammatical feature of all of these sentences is the copula is. We therefore refer to this fragment as *Cop*.

5.1.1 Semantic complexity

At the beginning of the previous section, we gave examples of valid and invalid arguments expressed using only Cop-sentences, and discussed their equivalence to unsatisfiable and satisfiable sets of Cop-sentences, respectively. The examples given were standard three-line syllogisms, but of course, there is no reason to impose any limit on length: the following argument is also valid.

Every stoic is a philosopher
Every philosopher is a mortal
Diogenes is a stoic
Diogenes is a mortal

Checking the validity of this example does, however, clearly take more steps of reasoning that were needed for the earlier arguments. What we would like to know is exactly how many more steps are required in general to decide validity

as the number of sentences is increased, and, in fact, whether it is even possible to decide the validity of some arguments.

More precisely, we wish to answer the following: given an argument expressed using only sentences of Cop, is there an algorithm for deciding its validity? Equivalently, since Cop contains sentence negation, is there an algorithm for deciding the satisfiability of a given set of Cop-sentences? If the answer is yes, what then is the computational complexity of the decision procedure?

The following result was proved in [58]:

Theorem 5.1. *The satisfiability of a set of sentences of Cop can be decided in deterministic polynomial time.*

Thus, deciding which sets of Cop-sentences represent logically-possible situations – equivalently, which syllogistic arguments are valid – is a tractable problem.

5.1.2 Expressive power

The preceding result allows us to find out whether a set of Cop-sentences is inconsistent, or whether it describes some possible situation. What it does not tell us, however, is anything about the *properties* of situations which consistent sets of Cop-sentences can describe. Given any pair of situations, under what circumstances can sentences of Cop tell them apart? We now attempt to answer this question.

As in the previous section, we proceed by translating sentences of Cop into formulae of first-order logic, and answering the corresponding questions applied to the formal language. Thus, when we say that a set of Cop-sentences describes a situation, we mean that there exists at least one structure interpreting the relevant first-order language in which the corresponding Cop-formulae are true, and when we speak of a Cop-sentence S distinguishing a pair of situations, we mean that for some given pair of structures $\mathfrak{A}$ and $\mathfrak{B}$, the semantics ϕ of S is true in one of $\mathfrak{A}$ or $\mathfrak{B}$ and false in the other. So, for example, if $\mathfrak{A}$ is defined over the domain $\{a\}$ by $\mathit{socrates}^{\mathfrak{A}} = a$, $\mathit{mortal}^{\mathfrak{A}} = \{a\}$, and $\mathfrak{B}$ is defined over the same domain by $\mathit{socrates}^{\mathfrak{B}} = a$, $\mathit{mortal}^{\mathfrak{B}} = \emptyset$, then the Cop-sentence Socrates is a mortal distinguishes $\mathfrak{A}$ and $\mathfrak{B}$.

Thus, to characterise the expressive power of Cop, we need to find some condition which pairs of structures must satisfy whenever *no* Cop-sentence can distinguish between them – i.e., when both structures make exactly the same

Cop-sentences true. Such a condition would then play the same role for Cop as bisimulation plays for modal logic.

Suppose that we are dealing with some fixed content lexicon, corresponding to a first-order signature $\Sigma = (C, P)$, where C is the set of constants generated from the proper nouns in the lexicon, and P is the set of unary predicates generated from the common nouns.

Definition 5.2. Let $\mathfrak{A}$ be any structure interpreting Σ. Let the *Cop-configuration of* $\mathfrak{A}$ be the function $\mathrm{conf}^{\mathfrak{A}} : P \times P \to \{0, 1, 2, 3, 4\}$ defined as follows for p, $q \in P$.

$$\begin{aligned}\mathrm{conf}^{\mathfrak{A}}(p, q) = {} & 0 \text{ if } p^{\mathfrak{A}} = q^{\mathfrak{A}} \\ & 1 \text{ if } p^{\mathfrak{A}} \subsetneq q^{\mathfrak{A}} \\ & 2 \text{ if } q^{\mathfrak{A}} \subsetneq p^{\mathfrak{A}} \\ & 3 \text{ if } p^{\mathfrak{A}} \cap q^{\mathfrak{A}} \neq \emptyset,\ q^{\mathfrak{A}} \setminus p^{\mathfrak{A}} \neq \emptyset \text{ and } p^{\mathfrak{A}} \setminus q^{\mathfrak{A}} \neq \emptyset \\ & 4 \text{ otherwise.}\end{aligned}$$

If $\mathfrak{A}$ and $\mathfrak{B}$ are structures interpreting S, we say that $\mathfrak{A}$ and $\mathfrak{B}$ are *Cop-similar* (written $\mathfrak{A} \sim_{\mathrm{Cop}} \mathfrak{B}$) if

1. $\mathrm{conf}^{\mathfrak{A}} = \mathrm{conf}^{\mathfrak{B}}$, and
2. for every constant $c \in C$, $\mathrm{tp}^{\mathfrak{A}}[c^{\mathfrak{A}}] = \mathrm{tp}^{\mathfrak{B}}[c^{\mathfrak{B}}]$.

The following theorem shows that Cop-similarity is precisely the notion we require to capture the expressive power of Cop. Recall that a Cop-sentence is true in $\mathfrak{A}$ if and only if its corresponding Cop-formula is true in $\mathfrak{A}$, and that the *Cop theory of* $\mathfrak{A}$ is the set of Cop-formulae over Σ which are true in $\mathfrak{A}$.

Theorem 5.3. *Let $\mathfrak{A}$ and $\mathfrak{B}$ be structures interpreting Σ. Then $\mathfrak{A} \sim_{Cop} \mathfrak{B}$ if and only if $\mathfrak{A}$ and $\mathfrak{B}$ have the same Cop theory (over the signature Σ.)*

Proof. Suppose that $\mathfrak{A} \sim_{\mathrm{Cop}} \mathfrak{B}$. We show that for every Cop-formula ϕ over Σ, if $\mathfrak{A} \models \phi$, then $\mathfrak{B} \models \phi$. Since Cop allows sentence negation, it then follows that if $\mathfrak{B} \models \phi$, then $\mathfrak{A} \models \phi$.

$\phi = \pm p(c)$ Immediate, since $\mathrm{tp}^{\mathfrak{A}}[c^{\mathfrak{A}}] = \mathrm{tp}^{\mathfrak{B}}[c^{\mathfrak{B}}]$.

$\phi = \forall x(p(x) \to q(x))$ If $\mathfrak{A} \models \phi$, then $p^{\mathfrak{A}} \subseteq q^{\mathfrak{A}}$, so $\mathrm{conf}^{\mathfrak{A}}(p, q) = 0$ or 1. Since $\mathfrak{A} \sim_{\mathrm{Cop}} \mathfrak{B}$, $\mathrm{conf}^{\mathfrak{B}}(p, q) = 0$ or 1, so $p^{\mathfrak{B}} \subseteq q^{\mathfrak{A}}$, and hence $\mathfrak{B} \models \phi$.

$\phi = \forall x(p(x) \to \neg q(x))$ If $\mathfrak{A} \models \phi$, then $p^{\mathfrak{A}} \cap q^{\mathfrak{A}} = \emptyset$, so $\mathrm{conf}^{\mathfrak{A}}(p, q) = 3$. Since $\mathfrak{A} \sim_{\mathrm{Cop}} \mathfrak{B}$, $\mathrm{conf}^{\mathfrak{B}}(p, q) = 3$, so $p^{\mathfrak{B}} \cap q^{\mathfrak{A}} = \emptyset$, and hence $\mathfrak{B} \models \phi$.

The other cases follow by symmetry of $\mathfrak{A}$ and $\mathfrak{B}$. Thus, if $\mathfrak{A} \sim_{\text{Cop}} \mathfrak{B}$, then $\mathfrak{A}$ and $\mathfrak{B}$ have the same Cop theory.

Conversely, suppose $\mathfrak{A} \not\sim_{\text{Cop}} \mathfrak{B}$. That is, suppose either that for some pair of unary predicates $p, q \in P$, $\text{conf}^{\mathfrak{A}}(p,q) \neq \text{conf}^{\mathfrak{B}}(p,q)$, or that for some constant c, $\text{tp}^{\mathfrak{A}}[c^{\mathfrak{A}}] \neq \text{tp}^{\mathfrak{B}}[c^{\mathfrak{B}}]$.

In the latter case, there exists some $p \in P$ such that $\mathfrak{A} \models p[c^{\mathfrak{A}}]$ and $\mathfrak{B} \not\models p[c^{\mathfrak{B}}]$, or vice versa, in which case the Cop formula $p(c)$ is true in one of $\mathfrak{A}$ and $\mathfrak{B}$, and false in the other.

For the former case, we show that for each possible pair of values of $\text{conf}^{\mathfrak{A}}(p,q)$ and $\text{conf}^{\mathfrak{B}}(p,q)$, we can construct a Cop formula true in $\mathfrak{A}$ and false in $\mathfrak{B}$. The proof is routine; for brevity, we describe only a single case here.

$\text{conf}^{\mathfrak{A}}(p,q) = 0$, $\text{conf}^{\mathfrak{B}}(p,q) = 1$**:** Then $p^{\mathfrak{A}} = q^{\mathfrak{A}}, p^{\mathfrak{B}} \subseteq q^{\mathfrak{B}}$, and $q^{\mathfrak{B}} \setminus p^{\mathfrak{B}} \neq \emptyset$, and so the Cop formula $\forall x(q(x) \to p(x))$ (and hence the Cop sentence Every q is a p) is true in $\mathfrak{A}$ and not $\mathfrak{B}$.

□

Thus the fragment Cop is capable of distinguishing between two structures if and only if they are Cop-dissimilar. As Cop-similarity is an easy condition to check, we can use it to prove inexpressibility results, such as the following.

Corollary 5.4. *Let P, Q, R be sets. No Cop-formula, or set of Cop-formulae, is equivalent to $P \cap R = Q \cap R$.*

Proof. Define two structures $\mathfrak{A}$ and $\mathfrak{B}$ over the domain $\{a_1, \ldots, a_5\}$, interpreting only p, q and r as follows

$$\begin{array}{ll} p^{\mathfrak{A}} = p^{\mathfrak{B}} = \{a_1, a_2\} & r^{\mathfrak{A}} = \{a_2, a_5\} \\ q^{\mathfrak{A}} = q^{\mathfrak{B}} = \{a_2, a_3, a_4\} & r^{\mathfrak{B}} = \{a_2, a_3, a_5\} \end{array}$$

and consider the Cop-fragment generated by the content lexicon consisting of common nouns corresponding to each of p, q and r. Neither $\mathfrak{A}$ nor $\mathfrak{B}$ interpret any constants, and it is easy to check that $\text{conf}^{\mathfrak{A}}$ and $\text{conf}^{\mathfrak{B}}$ have the same value for every combination of pairs from $\{p, q, r\}$, and so $\mathfrak{A} \sim_{\text{Cop}} \mathfrak{B}$, and by Theorem 5.3, $\mathfrak{A}$ and $\mathfrak{B}$ make the same Cop-sentences true. But $p^{\mathfrak{A}} \cap r^{\mathfrak{A}} = q^{\mathfrak{A}} \cap r^{\mathfrak{A}}$, and $p^{\mathfrak{B}} \cap r^{\mathfrak{B}} \neq q^{\mathfrak{B}} \cap r^{\mathfrak{B}}$, and so the result follows. □

Theorem 5.3 thus answers the question we posed earlier: under what circumstances do two structures make the same Cop-sentences true? The notion of Cop-simulation thus has a broadly similar role with regard to the Cop fragment of first-order logic as bisimulation has for the modal fragment. Recall also that bisimulation could be exploited to give a purely semantic characterisation of the modal fragment via the van Benthem Characterisation Theorem, given earlier as Theorem 3.28. Can we therefore prove an analogue of Theorem 3.28 with "Cop" replacing "modal fragment" and "Cop-simulation" replacing "bisimulation", thereby obtaining a semantic characterisation of Cop?

Unfortunately, the answer is: not quite.

Theorem 5.5. *There is no relation $\sim$ on structures such that for every first-order formula ϕ, ϕ is equivalent to a Cop-formula if and only if ϕ is invariant for $\sim$.*

Proof. Observe first that every Cop-formula is Horn. Let c be a constant, and p, q unary predicates, so that $p(c)$ and $q(c)$ are Cop-formulae, and let $\mathfrak{A}, \mathfrak{B}$ be structures interpreting at least those symbols such that $\mathfrak{A} \sim \mathfrak{B}$. Since $\sim$ preserves the truth of Cop-formulae, it follows that

$$\mathfrak{A} \models p[c^{\mathfrak{A}}] \text{ if and only if } \mathfrak{B} \models p[c^{\mathfrak{B}}]$$

and

$$\mathfrak{A} \models q[c^{\mathfrak{A}}] \text{ if and only if } \mathfrak{B} \models q[c^{\mathfrak{B}}].$$

But then it must also follow that

$$\mathfrak{A} \models p[c^{\mathfrak{A}}] \vee q[c^{\mathfrak{A}}] \text{ if and only if } \mathfrak{B} \models p[c^{\mathfrak{B}}] \vee q[c^{\mathfrak{B}}]$$

That is, $p(c) \vee q(c)$ is $\sim$-invariant. But $p(c) \vee q(c)$ is not a Horn formula, nor is it equivalent to any set of Horn formulae. To see this, recall from Theorem 3.47 that every satisfiable set Φ of Horn formulae has a unique "least" Herbrand model $\mathfrak{H}$ such that $\mathfrak{H} \models \bigwedge \Phi$, and every model $\mathfrak{A}$ of Φ satisfies every ground atom true in $\mathfrak{H}$. Now consider the Herbrand models $\mathfrak{H}$ defined by $p^{\mathfrak{H}} = \{c\}$ and $q^{\mathfrak{H}} = \emptyset$, and $\mathfrak{H}'$ defined by $p^{\mathfrak{H}'} = \emptyset$ and $q^{\mathfrak{H}'} = \{c\}$. Clearly, both $\mathfrak{H}$ and $\mathfrak{H}'$ are models of $p(c) \vee q(c)$, $p(c)$ is the only ground atom true in $\mathfrak{H}$, $q(c)$ is the only ground atom true in $\mathfrak{H}'$, and every model of $p(c) \vee q(c)$ must satisfy one of $p(c)$ or $q(c)$. The formula $p(c) \vee q(c)$ thus has no *unique* "least" Herbrand model, and so is not

equivalent to any set of Horn formulae. The result follows. □

Thus no semantic definition of Cop (or indeed of any Horn fragment) can be given in terms of invariance for some relation on structures. However, a somewhat weaker result is possible. Let Cop* be the fragment of English (and hence of logic) obtained by adding the following rules to the grammar of Cop:

$$\mathrm{IP}/\phi \wedge \psi \rightarrow \mathrm{IP}/\phi,\ \mathsf{and},\ \mathrm{IP}/\psi$$
$$\mathrm{IP}/\phi \vee \psi \rightarrow \mathrm{IP}/\phi,\ \mathsf{or},\ \mathrm{IP}/\psi$$

Since Cop contains sentence negation, Cop* is thus effectively the result of closing Cop under Boolean combinations of sentences.

Theorem 5.6. *A first-order formula ϕ is equivalent to a Cop*-formula if and only if ϕ is invariant for Cop-simulation.*

Proof. (Essentially the same as the proof of Theorem 3.28 – see, e.g., [7],[90].) Theorem 5.3 guarantees that any formula equivalent to a Cop-formula is invariant for $\sim_{\mathrm{Cop}}$, and so any formula equivalent to a Boolean combination of Cop-formulae (i.e., any Cop*-formula) must also be invariant for $\sim_{\mathrm{Cop}}$.

To show the converse, suppose that ϕ is invariant for Cop-simulation, and let

$$\Phi = \{\psi \mid \psi \text{ a Cop*-formula}, \phi \models \psi\}.$$

If we can show that $\Phi \models \phi$, then by compactness, there exists some finite subset $X \subseteq \Phi$ such that $\models \bigwedge X \rightarrow \phi$. By construction of Φ, $\models \phi \rightarrow \bigwedge X$, and so $\models \phi \leftrightarrow \bigwedge X$ – that is, ϕ is equivalent to a conjunction of Cop*-formulae, which must itself be a Cop*-formula.

Now, to show that $\Phi \models \phi$, suppose that Φ is consistent (otherwise we have trivially that $\Phi \models \phi$), and let $\mathfrak{A}$ be a model of Φ. Let

$$T = \{\psi \mid \psi \text{ a Cop*-formula and } \mathfrak{A} \models \psi\}.$$

We show that $T \cup \{\phi\}$ is consistent. For, suppose it was not. Then, by compactness, for some finite subset $X \subseteq T$, $\models \phi \rightarrow \neg \bigwedge X$. Now, by moving negations inward, we can see that $\neg \bigwedge X$ is logically equivalent to a Cop*-formula χ (since Cop* is closed under Boolean operators) which is a member of Φ. But then $\mathfrak{A} \models \neg \bigwedge X$, contradicting $X \subseteq T$, and $\mathfrak{A} \models T$. So $T \cup \{\phi\}$ is consistent.

Let $\mathfrak{B}$ be any model of $T \cup \{\phi\}$, and let ψ be any Cop*-formula. If $\mathfrak{A} \models \psi$, then ψ is an element of T, and so $\mathfrak{B} \models \psi$. Likewise, if $\mathfrak{A} \models \neg\psi$, then $\neg\psi$ is an element of T, and so $\mathfrak{B} \models \neg\psi$. Thus $\mathfrak{A}$ and $\mathfrak{B}$ have the same Cop* theory, and so the same Cop theory, and, by Theorem 5.3, $\mathfrak{A} \sim_{\text{Cop}} \mathfrak{B}$. Since $\mathfrak{B} \models \phi$ and ϕ is invariant for Cop-simulation, $\mathfrak{A} \models \phi$, and so $\Phi \models \phi$ as required. □

Note that the extra rules added for Cop* are far from logically harmless: in particular, it is easy to show, as we do in Chapter 6, that Cop* has an NP-complete satisfiability problem, compared to the PTIME result for Cop. However, as we saw, without this extra complexity, a result such as Theorem 5.6 is not possible.

5.2 Verbs

We now proceed to extend the grammar of Cop with new rules generating sentences containing transitive or ditransitive verbs such as admire and prefer, respectively. We thus obtain two fragments of English, Cop+TV and Cop+TV+DTV. Via the same process of semantic annotation as before, these define two fragments of logic which we can study in order to answer the same questions we posed for Cop, and so the semantic complexity and expressive power of each of these fragments can be compared.

Let Cop+TV be the fragment of English containing all sentences of Cop, as well as all sentences of the forms

$$\text{NP}_1 \left\{ \begin{array}{l} t\text{s} \\ \text{does not } t \end{array} \right\} \text{NP}_2$$

where t is a transitive verb and NP_1 and NP_2 are either proper nouns, or common nouns with the determiners every, some and no. Let Cop+TV+DTV be the fragment containing all sentences of Cop+TV, as well as those of the forms

$$\text{NP}_1 \left\{ \begin{array}{l} d\text{s} \\ \text{does not } d \end{array} \right\} \text{NP}_2 \text{ to } \text{NP}_3$$

where d is a ditransitive verb, and NP_1, NP_2 and NP_3 are as for Cop+TV. The assignment of semantics to each sentence is straightforward. Thus a Cop+TV

sentence such as Every man admires some mortal is interpreted by the formula

$$\forall x(man(x) \rightarrow \exists y(mortal(y) \wedge admire(x, y)))$$

whereas the Cop+TV+DTV sentence Every donkey prefers every carrot to every stick is interpreted as

$$\forall x(donkey(x) \rightarrow \forall y(carrot(y) \rightarrow \forall z(stick(z) \rightarrow prefer(x, y, z))))$$

Let TV denote the following set of phrase structure rules

Syntax	Formal Lexicon
$\mathrm{I}'/\phi \rightarrow \mathrm{VP}/\phi$	Neg $\rightarrow$ does not
$\mathrm{I}'/\phi \rightarrow \mathrm{NegP}/\phi$	
$\mathrm{NegP}/\neg\phi \rightarrow \mathrm{Neg}, \mathrm{VP}/\phi$	Content Lexicon
$\mathrm{VP}/\phi(\psi) \rightarrow \mathrm{TV}/\phi, \mathrm{NP}/\psi$	
	$\mathrm{TV}/\lambda s\lambda x[s(\lambda y[admire(x, y)])] \rightarrow$ admires
	$\mathrm{TV}/\lambda s\lambda x[s(\lambda y[despise(x, y)])] \rightarrow$ despises
	$\vdots$

and let DTV denote the rules

Syntax

$\mathrm{VP}/(\phi(\psi))(\pi) \rightarrow \mathrm{DTV}/\phi, \mathrm{NP}/\psi,$ to, NP/π

Content Lexicon

$\mathrm{DTV}/\lambda s\lambda t\lambda x[s(\lambda y[t(\lambda z[prefer(x, y, z)])])] \rightarrow$ prefers.

$\vdots$

A formal definition of Cop+TV is then given by the union of TV and the rules defining Cop, and of Cop+TV+DTV by the union of DTV and the rules defining Cop+TV. It is straightforward to verify that these rules generate precisely the sentences desired, bearing in mind that in order to keep the grammar simple,

we have ignored the surface syntactic issues of verb-inflection and the use of any instead of some in negative contexts. Neither of these details affect the semantics. Figures 5.2 and 5.3 show the derivation of IPs by the above grammar along with the simultaneous generation of their semantics.

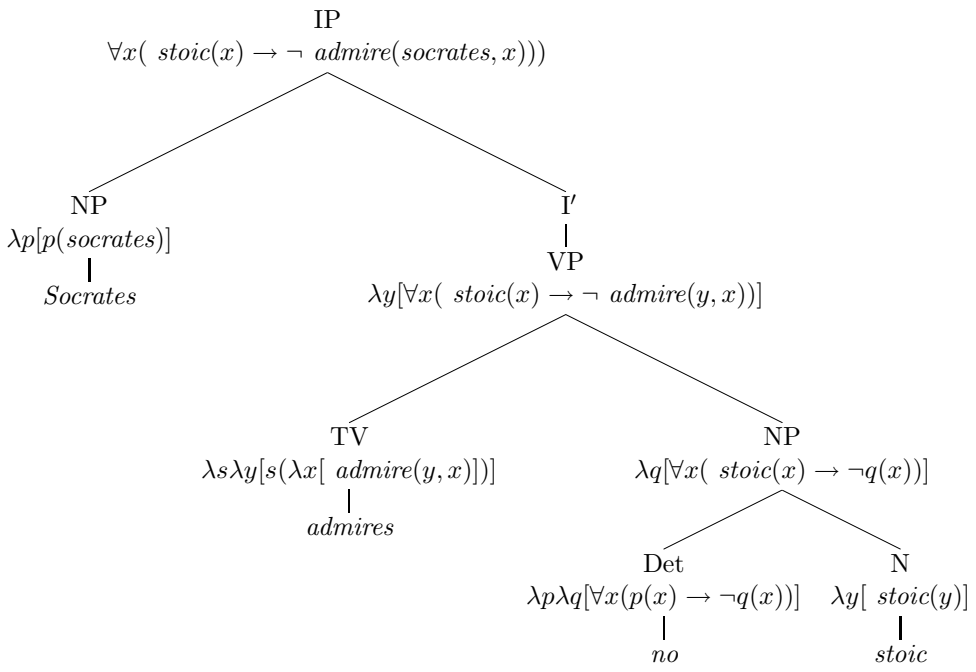


Figure 5.2: Structure of a simple Cop+TV sentence

The fragments of logic generated by Cop+TV and Cop+TV+DTV consist of all Cop-formulae, over the appropriate signature, and all formulae of the following forms

$$L_0$$
$$Q_1x_1(p_1(x_1) *_1 L_1)$$
$$Q_1x_1(p_1(x_1) *_1 Q_2x_2(p_2(x_2) *_2 L_2))$$
$$Q_1x_1(p_1(x_1) *_1 Q_2x_2(p_2(x_2) *_2 Q_3x_3(p_3(x_3) *_3 L_3)))$$

where, for $1 \leq i \leq 3$, p_i is a unary predicate, $(Q_i, *_i) \in \{(\forall, \rightarrow), (\exists, \wedge)\}$ and, for $0 \leq i \leq 3$, L_i is a non-unary literal involving *exactly* the variables $x_1, \ldots, x_i$, so that L_0 is ground. Note that the formulae unique to Cop+TV+DTV are precisely those containing ternary predicates.

In Cop, every sentence contains at most one determiner, and therefore every Cop-formula contains at most one quantifier. Cop sentences are thus entirely unambiguous with regard to relative quantifier scope. Sentences in Cop+TV, or

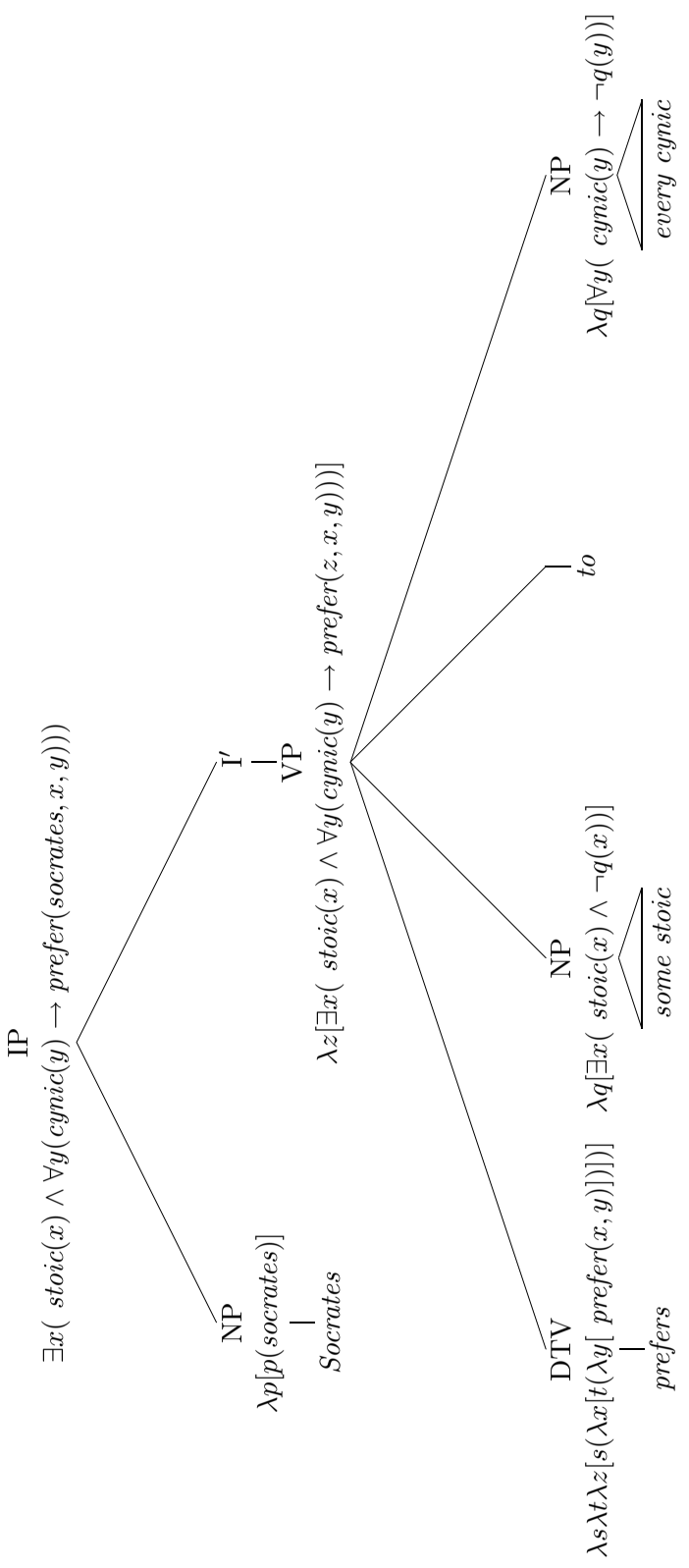


Figure 5.3: Structure of a simple Cop+TV+DTV sentence

Cop+TV+DTV, however, can contain up to three determiners, and hence the corresponding formulae can have multiple quantifiers. Each such sentence could thus in principle have several distinct readings. The by-now stereotypical example of a scope-ambiguous sentence is in fact a member of Cop+TV. The sentence

Every boy loves some girl

can have the readings

$$\forall x(boy(x) \rightarrow \exists y(girl(y) \wedge love(x, y)))$$

and

$$\exists x(girl(x) \wedge \forall y(boy(y) \rightarrow love(x, y)))$$

The rules given above generate only surface readings, such as the first of the preceding formulae. We could have complicated our grammar in such a way as to generate readings where the object of the sentence has wide scope, but it is simpler to make the following observation: if a Cop+TV sentence has a surface reading of the form $Q_1x(p(x) *_1 Q_2y(q(y) *_2 L_2))$, in the same notation as above, then it also has a reading of the form $Q_2x(q(x) *_2 Q_1y(p(y) *_1 L_2))$. A similar process of permutation generates the different readings of formulae containing three quantifiers. In terms of the forms of formulae generated, rescoping of sentences therefore has the effect of permuting the arguments to the non-unary predicate, and nothing else. So provided we take care, where possible, that our proofs do not rely on the order of such arguments, the addition of rescoping rules to the grammars of Cop+TV and Cop+TV+DTV has no effect on our results. For simplicity, we therefore assume initially that such rules are *not* present, and simply check this condition afterwards.

Clearly, since every Cop-sentence is also a sentence in both of the new fragments, anything expressible in Cop is also expressible in Cop+TV. Theorem 5.3 can be used to justify the intuitively obvious claim that the converse is false: the addition of transitive verbs constitutes a genuine increase in expressive power.

5.2.1 Semantic complexity

It might be assumed that the extra expressive power made available by the addition of transitive verbs would lead to an increase in the computational cost of determining satisfiability. In fact, it turns out that satisfiability in both Cop+TV

and Cop+TV+DTV is tractable, as we now show. These results were originally shown in [60], via a different proof to that given here, although the underlying ideas are the same.

Let E be an arbitrary set of Cop+TV+DTV-sentences, and let Φ be the set of Cop+TV+DTV-formulae representing the semantics of the elements of E, computed using the above grammar in polynomial time.

Skolemise Φ, and convert it into a set $\mathcal{C}_0$ of clauses equisatisfiable with Φ, and so by definition equisatisfiable with E. The elements of $\mathcal{C}_0$, written as implications for ease of reading, are of the forms

$\pm p_1(c)$	L_0
$p_1(x_1) \rightarrow \pm p_2(x_1)$	$p_1(x_1) \rightarrow L_1$
$p_1(x_1) \rightarrow p_2(f(x_1))$	$p_1(x_1) \wedge p_2(x_2) \rightarrow L_2$
$p_1(x_1) \wedge p_2(x_2) \rightarrow p_3(g(x_1, x_2))$	$p_1(x_1) \wedge p_2(x_2) \wedge p_3(x_3) \rightarrow L_3$

where c is a constant, p_1, p_2, p_3 are unary predicates, f, g are unary and binary (Skolem) function symbols, respectively and for each i, $0 \leq i \leq 3$, L_i is a non-unary literal containing exactly the variables $\{x_1, \ldots, x_i\}$. In order to reduce the number of cases which need to be considered, we rewrite $\mathcal{C}_0$ to give an equisatisfiable set $\mathcal{C}$ with the property that constant symbols only occur in unit clauses. To do this, for each constant c occurring in a non-unit clause $C \in \mathcal{C}_0$, let p_c be a fresh unary predicate symbol, let $C_1 = p_c(c)$ and $C_2 = \neg p_c(x) \vee C[x/c]$, where x is a variable not occurring in C. So if $C = \neg p_1(x) \vee t(x, c)$, then $C_1 = p_c(c)$ and $C_2 = \neg p_c(y) \vee \neg p_1(x) \vee t(x, y)$. Let $\mathcal{C}$ be the result of replacing each such C with the corresponding C_1 and C_2. It is straightforward to check that $\mathcal{C}_0$ and $\mathcal{C}$ are equisatisfiable.

Note that, unlike Cop, satisfiability in Cop+TV+DTV cannot be decided by unordered resolution: the presence of function symbols (introduced by Skolemisation of nested quantifiers) can lead to the generation of arbitrarily deep terms during resolution. For example, consider the Cop+TV+DTV sentences c is a p_1, Every p_1 t_1s some p_2, Every p_2 t_2s some p_3, Every p_3 is a p_1. The set of clauses generated from these sentences includes $p_1(c)$, $p_1(x_1) \rightarrow p_2(f_1(x_1))$, $p_2(x_2) \rightarrow p_3(f_2(x_2))$, $p_3(x_3) \rightarrow p_1(x_3)$. Unordered resolution can thus derive $p_3(f_2(f_1(c)))$, and hence, via the final clause $p_3(x_3) \rightarrow p_1(x_3)$, atoms $p_3(f_2(f_1(t)))$, where t is ground and of arbitrarily high depth.

Cop+TV+DTV is nonetheless decidable. To show this, we generate an equisatisfiable clause set $\mathcal{D}$ from $\mathcal{C}$, and show that the satisfiability of $\mathcal{D}$ can be

decided in PTIME by avoiding the generation of deep terms.

Let C be any clause in $\mathcal{C}$. Examination of the possible forms of C shows that all of the following properties hold.

P1. C is Horn.

P2. C involves at most one non-unary literal.

P3. If C involves a non-unary literal, then every unary literal of C has the form $\neg p(x)$, where p is a unary predicate and x is a variable.

P4. If a variable x appears in C, then x appears in some literal of the form $\neg p(x)$, where p is a unary predicate.

P5. If C involves a positive unary literal, then C has one of the forms

$$\begin{array}{ll} p_1(c) & p_1(x_1) \rightarrow p_2(x_1) \\ p_1(x_1) \rightarrow p_2(f(x_1)) & p_1(x_1) \wedge p_2(x_2) \rightarrow p_3(g(x_1, x_2)) \end{array}$$

where c is a constant, p_1, p_2, p_3 are unary predicates and f, g are (Skolem) function symbols.

Recall from Chapter 3 that the set of non-negative clauses in $\mathcal{C}$ is always satisfiable, and, by (P1), saturating these clauses under hyperresolution allows us to construct an Herbrand structure $\mathfrak{H}$ which is uniquely "minimal" with respect to $\mathcal{C}$ in the following sense: if $\mathcal{C}$ is satisfiable, then $\mathfrak{H} \models \mathcal{C}$, and if $\mathfrak{A}$ is *any* model of $\mathcal{C}$, then the set of ground atoms true in $\mathfrak{H}$ is a subset of the set of ground atoms true in $\mathfrak{A}$.

We begin by noting that the distribution of function symbols in $\mathcal{C}$ is in fact severely limited, a fact we can exploit in order to show decidability. Observe that every function symbol occurring in $\mathcal{C}$ was introduced by the Skolemisation of a single Cop+TV+DTV-formula in which existential quantifiers occurred within the scope of universal quantifiers. Each of these function symbols thus appears in exactly two clauses in $\mathcal{C}$, one of which, by (P5) is of the form $\gamma_f(\bar{x}) \rightarrow o_f(f(\bar{x}))$, where $\bar{x}$ is a tuple of variables of the same arity as f, o_f is a unary predicate and $\gamma_f(\bar{x})$ is a conjunction of atoms $p_{f,1}(x_1) \wedge \ldots \wedge p_{f,n}(x_n)$, where $\bar{x} = x_1, \ldots, x_n$ and for all $1 \leq i \leq n$, $p_{f,i}$ is a unary predicate. Note that both $\gamma_f(\bar{x})$ and o_f are uniquely determined by f. We introduce the following notation. If p, p' are unary predicates, write $p \Rightarrow p'$ if there exist predicates $p_0, \ldots, p_n$ for some n such that

$p = p_0$, $p' = p_n$ and the clauses $p_i(x) \rightarrow p_{i+1}(x)$ are in $\mathcal{C}$ for every i, $0 \leq i \leq n$. It follows from (P5) that for all unary predicates p, functions f and tuples of ground terms $\bar{t}$ of the appropriate length

$$\mathfrak{H} \models p(f(\bar{t})) \text{ iff } \mathfrak{H} \models \gamma_f(\bar{t}) \text{ and } o_f \Rightarrow p \tag{P6}$$

An immediate consequence of (P6) is that for any pair of tuples of ground terms $\bar{t}, \bar{t}'$ of the same arity as f such that $\mathfrak{H} \models \gamma_f(\bar{t}) \wedge \gamma_f(\bar{t}')$, it must be the case that $\text{tp}^{\mathfrak{H}}[f^{\mathfrak{H}}(\bar{t})] = \text{tp}^{\mathfrak{H}}[f^{\mathfrak{H}}(\bar{t}')]$

For every function symbol f occurring in $\mathcal{C}$, let c_f be a fresh constant symbol, let C'_f be the clause $\gamma_f(\bar{x}) \rightarrow o_f(c_f)$ and let $\mathcal{D} = \mathcal{C} \cup \{C'_f | C_f \in \mathcal{C}\}$. Note that $\mathcal{D}$ also satisfies properties (P1) - (P4), and a modified property (P5):

P5′. If C involves a positive unary literal, then C has one of the forms in (P5) or one of the forms

$$p_1(x_1) \rightarrow p_2(c_f) \quad p_1(x_1) \wedge p_2(x_2) \rightarrow p_3(c_g)$$

where c_f, c_g are constant symbols, p_1, p_2, p_3 are unary predicate symbols and f, g are (Skolem) function symbols.

As a consequence of (P1), (P4), (P5) and (P5′), hyperresolution applied to clauses in $\mathcal{D}$ can only derive ground unit clauses.

We now saturate $\mathcal{D}$ under hyperresolution, restricted so that no clause containing a term of depth greater than 3 is derived. Let $\mathcal{D}^*$ be the resulting set of clauses (which may, of course, contain $\perp$). By a similar argument to the one above, we can prove an analogue of (P6) for $\mathcal{D}^*$. For all unary predicates p, functions f and tuples of ground terms $\bar{t}$ of depth less than or equal to 2,

$$\begin{aligned} p(f(\bar{t})) \in \mathcal{D}^* \text{ iff } \gamma_f(\bar{t}) \in \mathcal{D}^*, o_f \Rightarrow p \text{ and } d(\bar{t}) \leq 2 \\ p(c_f) \in \mathcal{D}^* \text{ iff } \gamma_f(\bar{t}') \in \mathcal{D}^* \text{ for some } \bar{t}' \text{ and } o_f \Rightarrow p \end{aligned} \tag{P6′}$$

The intuition behind the construction of $\mathcal{D}$ and $\mathcal{D}^*$ is that, due to (P6), if we wish to decide which unary predicates hold of a given term t in $\mathfrak{H}$, all that matters is the leading function symbol of t, and not its functional depth. The

following lemma establishes that $\mathcal{D}^*$ contains enough information to decide the satisfiability of $\mathcal{C}$.

Lemma 5.7. *1. For every unary predicate p and function symbol f of arity n, there is an n-tuple $\bar{t}$ of ground terms such that $\mathfrak{H} \models p(f(t))$ if and only if $p(c_f) \in \mathcal{D}^*$.*

2. Let $p_1, \ldots, p_n$ be any sequence of unary predicates occurring in $\mathcal{C}$ (and hence in $\mathcal{D}$ and $\mathcal{D}^*$.) Then there is a ground term t such that $\mathfrak{H} \models \bigwedge_{i=1}^{n} p_i(t)$ if and only if there is a ground term t' such that for all $1 \leq i \leq n$, $p_i(t') \in \mathcal{D}^*$.

Proof. 1. Let $\bar{t} = t_1, \ldots, t_n$, suppose that

$$\gamma_f(x_1, \ldots, x_n) = p_{f,1}(x_1) \wedge \ldots \wedge p_{f,n}(x_n)$$

and suppose that $\mathfrak{H} \models p(f(\bar{t}))$. By (P6), $\mathfrak{H} \models \gamma_f(\bar{t})$ and $o_f \Rightarrow p$. If $d(t_i) \leq 3$ for all $1 \leq i \leq n$, we have by construction of $\mathcal{D}^*$ that $p_{f,i}(t_i) \in \mathcal{D}^*$ for all $1 \leq i \leq n$, and so by (P6), $p(c_f) \in \mathcal{D}^*$. Otherwise, suppose for induction that for some $1 \leq i \leq n$, $d(t_i) > 3$ and for all m-tuples $\bar{t'}$ of terms all of depth less than $d(t_i)$, functions g of arity m and predicates q, if $\mathfrak{H} \models q(g(\bar{t'}))$ then $q(c_g) \in \mathcal{D}^*$.

Since $d(t_i) > 3$, there is a function g of arity m and an m-tuple $\bar{t'}$ of ground terms such that $t_i = g(\bar{t'})$, $d(\bar{t'}) \geq 3$. So by (P6), $\mathfrak{H} \models \gamma_g(\bar{t'})$ and $o_f \Rightarrow p_{f,i}$. But then by the induction hypothesis, if

$$\gamma_g(x_1, \ldots, x_m) = p_{g,1}(x_1) \wedge \ldots \wedge p_{g,m}(x_m)$$

then $p_{g,j}(c_j) \in \mathcal{D}^*$ for all $1 \leq j \leq m$ and constants c_j. Since $\gamma_g(\bar{x}) \rightarrow o_g(c_g)$ is in $\mathcal{D}$ and $o_g \Rightarrow p_{f,i}$, it follows that $p_{f,i}(c_g) \in \mathcal{D}^*$. By repeating this process for every term in $\bar{t}$, we get a sequence $c_1, \ldots, c_n$ of constants such that $p_{f,i}(c_i) \in \mathcal{D}^*$ for all $1 \leq i \leq n$. Since $\gamma_f(\bar{x}) \rightarrow o_f(c_f)$ is in $\mathcal{D}$ and $o_f \Rightarrow p$, it follows that $p(c_f)$ is in $\mathcal{D}^*$ as required.

Conversely, suppose that $p(c_f) \in \mathcal{D}^*$. Then there exists a hyperresolution proof of $p(c_f)$ from $\mathcal{D}^*$ and by (P6′), $o_f \Rightarrow p$ and there is an n-tuple $\bar{t}$ of ground terms all of depth less than or equal to 3 such that $\gamma_f(\bar{t}) \in \mathcal{D}^*$. Suppose as before that $\gamma_f(\bar{x}) = p_{f,1}(x_1) \wedge \ldots \wedge p_{f,n}(x_n)$.

Let t_i be an element of $\bar{t}$. If t_i contains only constants and function symbols occurring in $\mathcal{C}$, then any hyperresolution proof of $p_{f,i}(t_i)$ from $\mathcal{D}$ is a proof of $p_{f,i}(t_i)$ from $\mathcal{C}$. Otherwise, t_i contains a subterm c_g, where g is a function symbol.

If $t_i = c_g$ for some function g of arity m then by (P6′), we know that $o_g \Rightarrow p_{f,i}$, $\gamma_g(\bar{x}) \rightarrow o_g(c_g) \in \mathcal{D}$ and for some m-tuple $\bar{t'}$ of ground terms, $\gamma_g(\bar{t'})$ can be proved from $\mathcal{D}$ by hyperresolution, and every element of $\bar{t'}$ has depth 3 or less. If every element of $\bar{t'}$ occurs in $\mathcal{C}$, then $\mathfrak{H} \models \gamma_g(\bar{t'})$ and, since we know that $\gamma_g(\bar{x}) \rightarrow o_g(c_g)$ is in $\mathcal{D}$, $\gamma_g(\bar{x}) \rightarrow o_g(g(\bar{x}))$ is in $\mathcal{C}$ and so $\mathfrak{H} \models o_g(g(\bar{t'}))$ and $\mathfrak{H} \models p_{f,i}(g(\bar{t'}))$. If some element of $\bar{t'}$ does *not* occur in $\mathcal{C}$, then proceed recursively. Since for every function g, c_g occurs in no purely positive clauses in $\mathcal{D}$, eventually such recursion terminates.

Finally, if $t_i = g(\bar{t'})$ for some function g of arity m and m-tuple $\bar{t'}$ of ground terms, then by (P6′), $\gamma_g(\bar{t'})$ can be proved from $\mathcal{D}$ by hyperresolution, every element of $\bar{t'}$ has depth 2 or less, and $o_f \Rightarrow p_{f,i}$. But then by (P6′), we know that $p_{f,i}(c_f) \in \mathcal{D}^*$ and the argument for the previous case applies.

Repeating the above argument for every t_i in $\bar{t}$ generates an n-tuple $\bar{t'}$ of terms in the universe of $\mathfrak{H}$ such that $\mathfrak{H} \models p(f(\bar{t}))$, as required.

2. Suppose that for some ground term t, $\mathfrak{H} \models \bigwedge_{i=1}^{n} p_i(t)$. If t is a constant, then by construction of $\mathcal{D}^*$, $p_i(t) \in \mathcal{D}^*$ for all $1 \leq i \leq n$. Otherwise, for some m-ary function symbol f and m-tuple of ground terms $\bar{t}$, $t = f(\bar{t})$. But then by case 1, $p_i(c_f) \in \mathcal{D}^*$ for all $1 \leq i \leq n$.

   Conversely, let t be a ground term such that for all $1 \leq i \leq n$, $p_i(t) \in \mathcal{D}^*$. If $t = c_f$ for some m-ary function f, then by case 1, there are n m-tuples of ground terms $\bar{t_1}, \ldots, \bar{t_n}$ such that $\mathfrak{H} \models \bigwedge_{i=1}^{n} p_i(f(\bar{t_i}))$. By (P6), for all $1 \leq i, j \leq n$, $\mathrm{tp}^{\mathfrak{H}}[f(t_i)] = \mathrm{tp}^{\mathfrak{H}}[f(t_j)]$ and so for any $1 \leq i \leq n$, $\mathfrak{H} \models \bigwedge_{j=1}^{n} p_j(f(t_i))$, as required. If $t = f(\bar{t'})$ for some m-ary function f and m-tuple of ground terms $\bar{t'}$, then by (P6′), $o_f \Rightarrow p$, $\gamma_f(\bar{t'})$ can be derived from $\mathcal{D}$ by hyperresolution and every element of $\bar{t'}$ has depth 2 or less. But then by (P6′) again, $p_i(c_f) \in \mathcal{D}^*$ for all $1 \leq i \leq n$, and the argument of the preceding paragraph applies. If $t = c$ for some constant c occurring in $\mathcal{C}$, then trivially $\mathfrak{H} \models \bigwedge_{i=1}^{n} p_i(c)$.

□

We are now in a position to show the decidability of Cop+TV+DTV.

Theorem 5.8. *The satisfiability of a set E of sentences of Cop+TV+DTV can be decided in deterministic polynomial time.*

Proof. Construct $\mathcal{C}$, and hence $\mathcal{D}$ and $\mathcal{D}^*$ as described above. The number of symbols in $\mathcal{D}^*$ is bounded by a polynomial of the size $\|E\|$ of E, and so the construction of $\mathcal{D}^*$ terminates in PTIME. It remains only to show that $\mathcal{C}$ is unsatisfiable if and only if $\bot \in \mathcal{D}^*$.

Suppose that $\mathcal{C}$ is unsatisfiable. Then there exists a hyperresolution proof $\mathbb{D}$ of $\bot$ from $\mathcal{C}$. Recall that $\mathfrak{H}$ is constructed by hyperresolution from the non-negative clauses of $\mathcal{C}$, and thus every clause derived by any *non*-final step of $\mathbb{D}$ holds in $\mathfrak{H}$. The final steps of $\mathbb{D}$ must be of the form

$$\dfrac{\begin{array}{cccc} \vdots & & \vdots & \\ m_1 & \ldots & m_j & \neg m'_1 \vee \ldots \vee \neg m'_j \vee \neg m \end{array} \qquad \dfrac{\begin{array}{cccc} \vdots & & \vdots & \\ l_1 & \ldots & l_i & \neg l'_1 \vee \ldots \vee \neg l'_i \vee l \end{array}}{l\theta}}{\bot}$$

where $i, j \in \mathbb{N}, l_1, \ldots, l_i$ and $m_1, \ldots, m_j, l$ are positive literals, $\neg l'_1, \ldots, \neg l'_i$ and $\neg m'_1, \ldots, \neg m'_j, \neg m$ are negative literals, $l_k\theta = l'_k\theta$ for $1 \leq k \leq i$, $m_k\theta' = m'_k\theta'$ for $1 \leq k \leq j$ and $(l\theta)\theta' = m\theta'$ for some substitutions θ and θ'. The clauses $C = \neg l'_1 \vee \ldots \vee \neg l'_i \vee l$ and $C' = \neg m'_1 \vee \ldots \neg m'_j \vee \neg m$ must, since, hyperresolution on Horn clause sets can only derive unit clauses, be members of $\mathcal{C}$ (and hence of $\mathcal{D}$.) By considering all of the possibilities for C and C', it can be seen that the set of unit clauses $l_1, \ldots, l_i, m_1, \ldots, m_j$ must be made up of collections of positive unit clauses such that for a tuple of ground terms $\bar{t}$, for each term t in $\bar{t}$, there is a set $p^t_1, \ldots, p^t_n$ of unary predicates such that $\mathfrak{H} \models \bigwedge_{i=1}^{n} p^t_i(t)$ and for some possibly empty tuples $\bar{t_1}, \ldots, \bar{t_m}$ of terms selected from $\bar{t}$, functions $f_1, \ldots, f_m$ and predicates $q_1, \ldots, q_m$, $\mathfrak{H} \models \bigwedge_{i=1}^{m} q_i(f_i(\bar{t_i}))$. Lemma 5.7 and (P6′) then guarantee that $\bot \in \mathcal{D}^*$.

Conversely, if $\bot \in \mathcal{D}^*$, then there exists a proof $\mathbb{D}$ of $\bot$ from $\mathcal{D}$ by hyperresolution. The final steps in $\mathbb{D}$ must be of the form given above, save that the non-unit clauses C and C' (using the same notation as before) must be elements of $\mathcal{D}$, but are not necessarily in $\mathcal{C}$. For all of the cases where C and C' *are* both in $\mathcal{C}$, Lemma 5.7 guarantees, via a similar argument to the one above, that $\bot$ can be derived from $\mathcal{C}$. By (P5′), the only other possibility is that the final steps of $\mathbb{D}$ are of the form

$$\dfrac{\dfrac{\vdots}{p(x) \to \neg o_f(x) \quad p(c_f)} \quad \dfrac{\dfrac{\vdots}{\gamma_f(\bar{t})} \quad \gamma_f(\bar{x}) \to o_f(c_f)}{o_f(c_f)}}{\bot}$$

where f is an n-ary function symbol. Since $p(c_f)$ and $\gamma_f(\bar{t})$ must be elements of $\mathcal{D}^*$, we know that by (P6′) that $o_f \Rightarrow p$ and by Lemma 5.7, there exists a tuple $\bar{t'}$ of ground terms such that $\mathfrak{H} \models \gamma_f(\bar{t'})$, so that by (P6), $\mathfrak{H} \models p(f(\bar{t'}))$. Hence $\bot$ can be derived from $\mathcal{C}$ via a hyperresolution derivation terminating with the following steps

$$\dfrac{\dfrac{\vdots}{p(x) \to \neg o_f(x) \quad p(f(\bar{t'}))} \quad \dfrac{\dfrac{\vdots}{\gamma_f(\bar{t'})} \quad \gamma_f(\bar{x}) \to o_f(f(\bar{x}))}{o_f(f(\bar{t'}))}}{\bot}.$$

Thus $\mathcal{C}$ is unsatisfiable, completing the proof. □

In essence, the preceding proof works because of the limited ability of Cop+TV and Cop+TV+DTV to describe NPs: without relative clauses, all that can be known is how the denotation of a *single* unary predicate is related to the denotations of at most a pair of unary predicates. No reference can be made within a Cop+TV(+DTV)-sentence to more than one predicate satisfied by an individual, or more than one relation in which it stands. In fact, the relations themselves – the verbs – play a very small role in the proof of Theorem 5.8. In particular, the proof made no reference to the order of arguments to non-unary relations, and thus, as we noted earlier, Theorem 5.8 continues to hold even when we add rules for quantifier rescoping to Cop+TV and Cop+TV+DTV.

5.2.2 Expressive power

The fact that fragments can have different abilities to distinguish situations and yet have the same semantic complexity demonstrates that complexity is too coarse to serve as a characterisation of expressive power. To find a better one, let us begin with Cop+TV: as in the case of Cop, we define a relation between structures preserving truth of Cop+TV-sentences. Let $\Sigma = (C, P, T)$ be a signature consisting of constants C, unary predicates P and binary relations T.

Definition 5.9. Let $\mathfrak{A}$ be a structure interpreting Σ. Let the *TV-configuration of* $\mathfrak{A}$ *over* Σ be the function $\mathrm{tv}^{\mathfrak{A}} : P \times P \times T \to \{1, 2, 3, 4, 5, 6\}$ defined as follows

for $p, q \in P$, $t \in T$.

$$\mathrm{tv}^{\mathfrak{A}}(p,q,t) = \begin{array}{ll} 1 & \text{if } (p^{\mathfrak{A}} \times q^{\mathfrak{A}}) \cap t^{\mathfrak{A}} = \emptyset \\ 2 & \text{if } p^{\mathfrak{A}} \times q^{\mathfrak{A}} \subseteq t^{\mathfrak{A}} \text{ and } p^{\mathfrak{A}} \times q^{\mathfrak{A}} \neq \emptyset \\ 3 & \text{if } p^{\mathfrak{A}} \neq \emptyset \text{ and for every } a_1 \in p^{\mathfrak{A}}, \text{ there exist} \\ & a_2, a_3 \in q^{\mathfrak{A}} \text{ such that } (a_1, a_2) \in t^{\mathfrak{A}}, (a_1, a_3) \notin t^{\mathfrak{A}} \\ 4 & \text{if } q^{\mathfrak{A}} \neq \emptyset, \text{ there exists } a_1 \in p^{\mathfrak{A}} \text{ such that} \\ & (\{a_1\} \times q^{\mathfrak{A}}) \cap t^{\mathfrak{A}} = \emptyset \text{ and there exists } a_2 \in p^{\mathfrak{A}} \\ & \text{such that } \{a_2\} \times q^{\mathfrak{A}} \subseteq t^{\mathfrak{A}} \\ 5 & \text{if } (p^{\mathfrak{A}} \times q^{\mathfrak{A}}) \cap t^{\mathfrak{A}} \neq \emptyset, \text{ there exists } a_1 \in p^{\mathfrak{A}} \text{ such} \\ & \text{that } (\{a_1\} \times q^{\mathfrak{A}}) \cap t^{\mathfrak{A}} = \emptyset \text{ and for every } a_2 \in p^{\mathfrak{A}}, \\ & \text{there exists } a_3 \in q^{\mathfrak{A}} \text{ such that } (a_2, a_3) \notin t^{\mathfrak{A}} \\ 6 & \text{otherwise} \end{array}$$

If $\mathfrak{A}$ and $\mathfrak{B}$ are structures interpreting Σ, we say that $\mathfrak{A}$ and $\mathfrak{B}$ are *TV-similar* ($\mathfrak{A} \sim_{\mathrm{TV}} \mathfrak{B}$) if

1. $\text{tv-conf}^{\mathfrak{A}} = \text{tv-conf}^{\mathfrak{B}}$,
2. for every pair of constants c, d, $\mathrm{tp}^{\mathfrak{A}}[c^{\mathfrak{A}}, d^{\mathfrak{A}}] = \mathrm{tp}^{\mathfrak{B}}[c^{\mathfrak{B}}, d^{\mathfrak{B}}]$,
3. for every $c \in C$, $p \in P$ and $t \in T$:
   (a) there exists $a \in A$ such that $\mathfrak{A} \models p[a] \wedge t[a, c^{\mathfrak{A}}]$ iff there exists $b \in B$ such that $\mathfrak{B} \models p[b] \wedge t[b, c^{\mathfrak{B}}]$,
   (b) there exists $a \in A$ such that $\mathfrak{A} \models p[a] \wedge t[c^{\mathfrak{A}}, a]$ iff there exists $b \in B$ such that $\mathfrak{B} \models p[b] \wedge t[c^{\mathfrak{B}}, b]$, and
4. $\mathfrak{A} \sim_{\mathrm{Cop}} \mathfrak{B}$.

Each Cop+TV-formula can contain up to two unary predicates and one binary relation. The different cases in the definition of TV-configuration exhaust the possibilities of how sets of Cop+TV-sentences can describe the interactions of their denotations. Each case therefore corresponds to a particular sentence or set of sentences. Let p and q be common nouns (and hence unary predicates), and t a transitive verb (and hence binary relation.) For some structure $\mathfrak{A}$ interpreting p, q and t, the value of $\mathrm{tv}^{\mathfrak{A}}(p, q, t)$ corresponds to the sentences below. It is easily checked that every structure $\mathfrak{A}$ interpreting p, q and t satisfies exactly one of these possibilities.

1. No p *t*s any q.

2. Every p ts every q, Some p is a p, Some q is a q.

3. Every p ts some q, Every p does not t some q, Some p is a p.

4. Some p ts no q, Some p ts every q, Some q is a q.

5. Some p ts some q, Some p ts no q, No p ts every q.

6. Every p ts some q, Some p ts every q, Some p does not t some q.

Theorem 5.10. *Let $\mathfrak{A}$ and $\mathfrak{B}$ be structures interpreting $\Sigma = (C, P, T)$. Then $\mathfrak{A} \sim_{TV} \mathfrak{B}$ if and only if $\mathfrak{A}$ and $\mathfrak{B}$ have the same Cop+TV theory (over the signature Σ.)*

Proof. Essentially the same as that for Theorem 5.3. □

The definitions of Cop and TV configuration allowed us to prove Theorems 5.3 and 5.10, respectively, because both fragments are essentially finite. None of the phrase structure rules in their definitions are recursive, and so in each case there is a (small) upper bound on sentence length. This bound means that exhaustive lists of the possibilities for interpretation (as Definitions 5.2 and 5.9 are) can be used to characterise expressive power. The introduction of recursive phrase structure rules (in the form of relative clauses) in the following section rules out this approach for later fragments.

The following corollary to Theorem 5.10 shows that Cop+TV shares some of the limitations of the expressive power of Cop.

Corollary 5.11. *Let P, Q, R be sets. No Cop+TV-formula, or set of Cop+TV-formulae, is equivalent to $P \cap R = Q \cap R$.*

Proof. Define two structures, $\mathfrak{A}$ and $\mathfrak{B}$ over the domain $\{a_1, \ldots, a_5\}$ interpreting the unary predicates p, q, r and the binary predicate t as follows

$$\begin{array}{ll} p^{\mathfrak{A}} = p^{\mathfrak{B}} = \{a_1, a_2\} & r^{\mathfrak{A}} = \{a_2, a_5\} \\ q^{\mathfrak{A}} = q^{\mathfrak{B}} = \{a_2, a_3, a_4\} & r^{\mathfrak{B}} = \{a_2, a_3, a_5\} \\ t^{\mathfrak{A}} = t^{\mathfrak{B}} = \{(a_1, a_2)\} & \end{array}$$

It is straightforward to check that $\mathfrak{A}$ and $\mathfrak{B}$ are TV-similar, and thus by Theorem 5.10 make all the same Cop+TV-sentences true. But $p^{\mathfrak{A}} \cap r^{\mathfrak{A}} = q^{\mathfrak{A}} \cap r^{\mathfrak{A}}$, and $p^{\mathfrak{B}} \cap r^{\mathfrak{B}} \neq q^{\mathfrak{B}} \cap r^{\mathfrak{B}}$, and so the result follows. □

Since the Cop-formulae in the proof of Theorem 5.5 are also in Cop+TV, and every Cop+TV-formula is Horn, an analogue of Theorem 5.5 holds, and so in order to prove any kind of Invariance Theorem, we are forced to add sentence (IP) coordination via the same phrase-structure rules as earlier, to produce a fragment Cop+TV* closed under Boolean combinations of sentences. The following result then follows by a near identical argument to that for Theorem 5.6.

Theorem 5.12. *A first-order formula ϕ is equivalent to a Cop+TV*-formula if and only if ϕ is invariant for TV-simulation.*

As before, the addition of sentence coordination leads to an increase in complexity: Cop+TV* has an NP-complete satisfiability problem. This result is proved in Chapter 6.

There remains, of course, the question of the logical expressive power of Cop+TV+DTV, and of the effect of allowing quantifier rescoping in Cop+TV and Cop+TV+DTV. (The proof of Theorem 5.8 certainly is sensitive to the order of arguments to non-unary predicates.) None of these fragments are recursively defined, and so they all admit a finite characterisation.

We could proceed as we have done so far, and manually enumerate the options for, say, a Cop+TV+DTV configuration. However, perhaps this would be missing a trick. All of these fragments have had a tractable decision problem, and, for a given set of symbols from the content lexicon, only a very small number of ways in which all of these symbols can be combined to form grammatical sentences. We can take advantage of these facts in order to automate the generation of a suitable relation between structures.

Let F be any fragment satisfying the following conditions.

1. F is decidable.

2. Every F-sentence is an instance of one of only a finite number of sentence-schemata.

It is straightforward, given these conditions, to determine automatically a set $\mathcal{F}$ of sets of sentence schemata such that over a given signature Σ, every maximal consistent set of F-sentences over Σ is an instance of some member of $\mathcal{F}$. The notions of Cop- and TV-configuration essentially define such an $\mathcal{F}$.

This idea is perhaps best illustrated with an example. Let F_{ex} be the very small fragment defined by the grammar of Cop *without* the rules for proper nouns.

F_{ex} is clearly decidable, since Cop is, and every F_{ex}-sentence is an instance of one of the schemata

$$\left\{\begin{array}{l}\text{Every}\\ \text{Some}\\ \text{No}\end{array}\right\}\ p_1\ \left\{\begin{array}{l}\text{is}\\ \text{is not}\end{array}\right\}\ \text{a}\ p_2$$

where p_1, p_2 are common nouns.

Let $\mathcal{F}_{ex}$ consist of the union of the following cases, C1 to C4.

C1. Every p is a q, no p is a q (i.e., there are no ps.)

C2. Every p is a q, some p is a q.

C3. Some p is not a q, no p is a q.

C4. Some p is not a q, some p is a q.

It is straightforward to check that for any pair of common nouns p' and q', the maximal consistent set of F_{ex}-sentences containing both p' and q' is an instance of a member of $\mathcal{F}_{ex}$. Thus, if we take a common noun p, any structure interpreting p must make exactly one of the following cases true. These possibilities can be calculated directly from $\mathcal{F}_{ex}$ by substitution, discarding any results which are not satisfiable.

D1'. Every p is a p, no p is a p (i.e., there are no ps.)

D2'. Every p is a p, some p is a p (i.e., there are ps.)

For any fragment F satisfying conditions 1 and 2, and any structure $\mathfrak{A}$ interpreting the signature of F, the corresponding $\mathcal{F}$ can be used to compute an exhaustive list of the possible ways $\mathfrak{A}$ can interpret a given set of symbols. We can use this information to define a notion of simulation $\sim_F$ between structures such that for any structures $\mathfrak{A}$ and $\mathfrak{B}$ interpreting the same signature, $\mathfrak{A} \sim_F \mathfrak{B}$ if and only if $\mathfrak{A}$ and $\mathfrak{B}$ have the same F-theory.

It is thus possible to automate the construction of an appropriate relation of simulation for certain fragments, among them Cop+TV+DTV and the extensions of Cop+TV and Cop+TV+DTV with quantifier rescoping. Of course, as fragments become more complicated, the number of cases in the definition of simulation becomes correspondingly larger, but this process nonetheless greatly reduces the effort required of a human scholar in studying such fragments.

For Horn fragments such as Cop+TV(+DTV), both with and without quantifier rescoping, Theorem 5.5 continues to prevent the possibility of Invariance Theorems for those fragments alone. In fact, as we see in Section 5.3, many non-Horn fragments also fall foul of Theorem 5.5 too. However, just as earlier, the addition of sentence coordination to such fragments allows Invariance Theorems to be established.

Theorem 5.13. *Let F be a fragment satisfying conditions 1 and 2, let $\sim_F$ be an F-simulation, as computed above, and suppose that an analogue of Theorem 5.5 holds for F. Let F^* be the result of extending the grammar of F with rules for coordination of IP. Then an arbitrary first-order formula ϕ is equivalent to an F^*-formula if and only if ϕ is invariant for F-simulation.*

Proof. See the proof of Theorem 5.6. □

5.3 Relative Clauses

The preceding section considered only very small extensions to Cop, granting limited powers to describe relations between individuals, but sharing Cop's inability to say much about those individuals themselves. In this section, we take a different tack, returning to Cop and extending its ability to describe entities, with restrictive relative clauses.

Noun phrases in Cop are all either proper nouns such as Socrates, or quantified common nouns such as every man. With the addition of relative clauses, we gain the ability to embed Cop-sentences qualifying common nouns. So rather than just some philosopher, we can refer to some philosopher who is not a stoic, and limit the denotation of the whole NP. Relative clauses can of course be nested: some man who is a philosopher who is not a stoic is a perfectly acceptable English NP.

The interpretation of relative clauses such as these is uncontroversial: following [25], among others, we interpret them conjunctively. If some individual ι is a stoic who is not a cynic, then ι is a stoic *and* ι is not a cynic.

Let Cop+Rel be the fragment of English containing all sentences of the form NP_1 is (not) an NP_2, where NP_1 and NP_2 are noun phrases. A noun phrase in Cop+Rel is either a proper noun, or a quantified N' of the form every/some/no ϕ. An N' in Cop+Rel is either a common noun p, or has the form p who is (not) a ϕ (where p is a common noun and ϕ is an N'). So, for example, Every man who is a

stoic is a philosopher who is not a cynic is a Cop+Rel-sentence, as is No man who is a stoic who is not a cynic is an epicurean. With the conjunctive semantics outlined above, these are interpreted as

$$\forall x(man(x) \wedge stoic(x) \rightarrow philosopher(x))$$

and

$$\forall x(man(x) \wedge stoic(x) \wedge \neg cynic(x) \rightarrow epicurean(x))$$

respectively.

More formally, Cop+Rel is generated by the union of the following rules with those of Cop.

Syntax	Formal lexicon
$\mathrm{N}'/\phi(\psi) \rightarrow \mathrm{N}/\psi$, CP/ϕ	$\mathrm{C} \rightarrow$
$\mathrm{CP}/\phi(\psi) \rightarrow \mathrm{CSpec}_t/\phi$, C'_t/ψ	$\mathrm{RelPro}/\lambda q \lambda p \lambda x[p(x) \wedge q(x)] \rightarrow$ who,
$\mathrm{C}'_t/\lambda t[\phi] \rightarrow \mathrm{C}$, IP/ϕ	which
$\mathrm{NP}/\phi \rightarrow \mathrm{RelPro}/\phi$	
$\mathrm{CSpec}_t \rightarrow$	

In order to produce the correct semantics, wh-movement must be applied to sentences (IPs) generated by these rules, as outlined in Chapter 4, Section 4.4.1. In particular, we insist that: (i) every RelPro moves into the nearest CSpec_t which c-commands it; (ii) every CSpec_t is filled by a moved RelPro; and (iii) every NP position vacated by a RelPro moving to CSpec_t is filled by a trace t with semantic value $\lambda p.p(t)$. Figure 5.4 shows the generation of a sentence by the rules for Cop+Rel, with wh-movement illustrated, as well as the generation of the semantics.

In fact, a simple fragment such as this one could be specified just as precisely without any reference to wh-movement. For the sake of extensibility, however, we prefer the more general approach.

In what follows, we use who and which as relevant; the issue of animacy agreement between relative pronouns and their antecedents does not affect the semantics.

Note that since each Cop+Rel-sentence contains at most one determiner, each Cop+Rel-formula contains at most one quantifier. Unlike in Section 5.2, we therefore do not need to worry about quantifier scope.

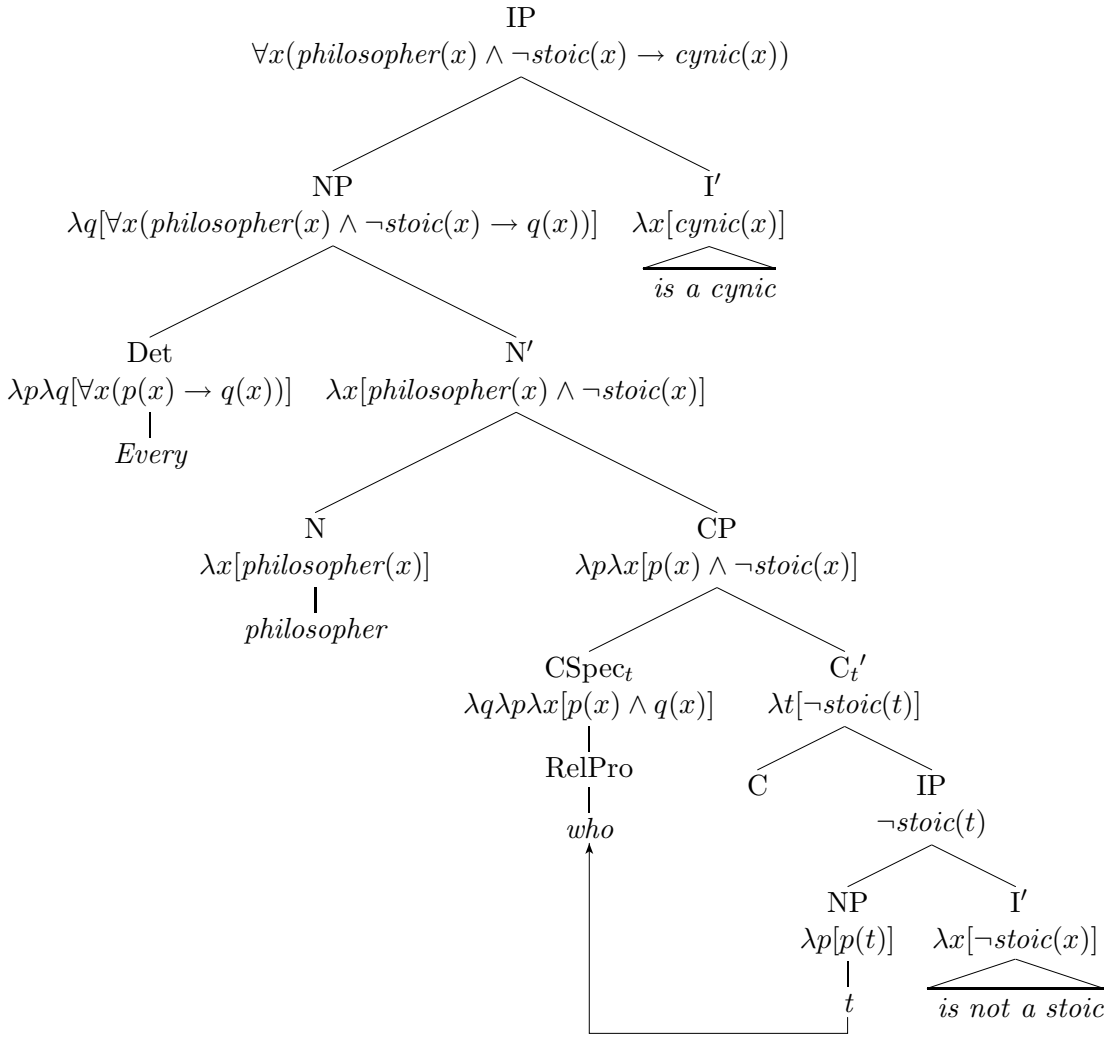


Figure 5.4: Structure of a simple Cop+Rel sentence

We define the fragment of logic generated by Cop+Rel in two stages. First, define an *N'-formula* recursively:

1. $p(x)$ is a an N'-formula (where p is a unary predicate.)
2. If $\pi(x)$ is an N'-formula, then $p(x) \wedge \pi(x)$ and $p(x) \wedge \neg\pi(x)$ are N'-formulae.

A *Cop+Rel-formula* is then of one of the forms

$$\pm p(c)$$

$$\forall x(\phi_1(x) \rightarrow \pm\phi_2(x))$$

$$\exists x(\phi_1(x) \wedge \pm\phi_2(x))$$

where c is a constant, p is a unary predicate and $\phi_1(x)$ and $\phi_2(x)$ are N'-formulae.

It is straightforward to verify that the grammar presented earlier does in fact generate all and only formulae of the above forms over the relevant signature. The N'-formulae, as the name suggests, are the subformulae of Cop+Rel-formulae corresponding to the semantics of N'-phrases.

Since Cop+Rel contains Cop as a subfragment, we know that every pair of Cop-distinguishable situations can also be distinguished by Cop+Rel. The following result shows that Cop+Rel is a genuine extension of Cop.

Proposition 5.14. *1. Cop+Rel is strictly more expressive than Cop.*

2. Cop+Rel can distinguish situations which cannot be distinguished by sentences of Cop+TV.

Proof. 1. Let $\mathfrak{A}$, $\mathfrak{B}$ be as in the proof of Corollary 5.11. That is, both structures are over the domain $\{a_1, \ldots, a_5\}$ and interpret unary predicates p, q and r and binary predicate t as follows

$$\begin{array}{ll} p^{\mathfrak{A}} = p^{\mathfrak{B}} = \{a_1, a_2\} & r^{\mathfrak{A}} = \{a_2, a_5\} \\ q^{\mathfrak{A}} = q^{\mathfrak{B}} = \{a_2, a_3, a_4\} & r^{\mathfrak{B}} = \{a_2, a_3, a_5\} \\ t^{\mathfrak{A}} = t^{\mathfrak{B}} = \{(a_1, a_2)\} & \end{array}$$

As noted before, $\mathfrak{A}$ and $\mathfrak{B}$ are Cop-similar, and so by Theorem 5.3, make all the same Cop-sentences true. However, the Cop+Rel-sentence Every q who is an r is a p, which translates to $\forall x(q(x) \wedge r(x) \rightarrow p(x))$, is true in $\mathfrak{A}$ and not in $\mathfrak{B}$.

2. It is relatively easy to show that $\mathfrak{A}$ and $\mathfrak{B}$ above are also Cop+TV-similar, whence the result follows by Theorem 5.10.

□

Incidentally, recall that Corollaries 5.4 and 5.11 showed that neither Cop nor Cop+TV could express $P \cap R = Q \cap R$, where P, Q and R are sets. It ought to be trivially clear that in any model $\mathfrak{A}$ of the Cop+Rel-sentences Every p who is an r is a q and Every q who is an r is a p, it must be the case that $p^{\mathfrak{A}} \cap r^{\mathfrak{A}} = q^{\mathfrak{A}} \cap r^{\mathfrak{A}}$.

5.3.1 Semantic Complexity

The main feature distinguishing Cop+Rel from the two earlier fragments is the ability to embed sentences recursively inside others, and it is this feature which is responsible for the extra complexity of deciding Cop+Rel. The following result was proved in [58]:

Theorem 5.15. *The satisfiability problem for Cop+Rel is NP-complete.*

5.3.2 Expressive Power

With the presence of recursive rules in the grammar of Cop+Rel, a simple enumeration of the situations the fragment can describe is clearly not possible as it was for the preceding fragments. In fact, the characterisation of expressive power for Cop+Rel is a great deal simpler for it.

As before, we now define a relation on structures corresponding to preservation of truth in Cop+Rel. Suppose again that $\Sigma = (C, P)$ is a signature of constants C and unary predicates P.

Definition 5.16. Let $\mathfrak{A}$ and $\mathfrak{B}$ be structures interpreting Σ. We say that $\mathfrak{A}$ and $\mathfrak{B}$ are *Cop+Rel-similar*, written $\mathfrak{A} \sim_{\mathrm{Rel}} \mathfrak{B}$, if

1. for every constant $c \in C$, $\mathrm{tp}^{\mathfrak{A}}[c^{\mathfrak{A}}] = \mathrm{tp}^{\mathfrak{B}}[c^{\mathfrak{B}}]$,

2. for every $a \in A$ satisfying a unary predicate, there exists $b \in B$ such that $\mathrm{tp}^{\mathfrak{A}}[a] = \mathrm{tp}^{\mathfrak{B}}[b]$.

3. for every $b \in B$ satisfying a unary predicate, there exists $a \in A$ such that $\mathrm{tp}^{\mathfrak{A}}[a] = \mathrm{tp}^{\mathfrak{B}}[b]$.

Theorem 5.17. *Let $\mathfrak{A}$ and $\mathfrak{B}$ be structures interpreting Σ. Then $\mathfrak{A} \sim_{Rel} \mathfrak{B}$ if and only if $\mathfrak{A}$ and $\mathfrak{B}$ have the same Cop+Rel theory over Σ.*

Proof. Suppose that $\mathfrak{A} \sim_{\text{Rel}} \mathfrak{B}$. We show that for every Cop+Rel formula ϕ, if $\mathfrak{A} \models \phi$, then $\mathfrak{B} \models \phi$. Since Cop+Rel contains sentence negation, it follows that if $\mathfrak{B} \models \phi$, then $\mathfrak{A} \models \phi$.

The truth of all ground Cop+Rel-formulae is evidently preserved by clause (2) in Definition 5.16. Each Cop+Rel-formula *not* containing a constant is of the form $\forall x(\phi_1(x) \rightarrow \pm\phi_2(x))$ or $\exists x(\phi_1(x) \wedge \pm\phi_2(x))$, where $\phi_1(x)$ and $\phi_2(x)$ are N′-formulae. A simple induction on the structure of N′-formulae suffices to show that $\mathfrak{A}$ and $\mathfrak{B}$ agree on the truth value of every Cop+Rel-formula.

Conversely, suppose now that $\mathfrak{A} \not\sim_{\text{Rel}} \mathfrak{B}$. Then one of (1), (2) or (3) in Definition 5.16 fails. We consider each case in turn.

1. For some constant c, $\text{tp}^{\mathfrak{A}}[c^{\mathfrak{A}}] \neq \text{tp}^{\mathfrak{B}}[c^{\mathfrak{B}}]$. As in the proof of Theorem 5.3, there exists a ground Cop-formula (and hence Cop+Rel-formula) ϕ such that $\mathfrak{A} \models \phi$ and $\mathfrak{B} \not\models \phi$.

2. There exists $a \in A$ satisfying some unary predicate such that for every $b \in B$, $\text{tp}^{\mathfrak{A}}[a] \neq \text{tp}^{\mathfrak{B}}[b]$. So for every $b \in B$, there exists a unary predicate p_b such that $\mathfrak{A} \models p_b[a]$ iff $\mathfrak{B} \models \neg p_b[b]$. Let $P_a^+ = \{p^+ \in P | \mathfrak{A} \models p^+[a]\}$, and let $P_a^- = P \setminus P_a^+$. We know that $P_a^+ \neq \emptyset$, since a satisfies some unary predicate p_1, and so the sentence

   $$\textsf{Some } p_1 \textsf{ \{ who is a } p^+ \textsf{ \}}_{p^+ \in P_a^+} \textsf{ \{ who is not a } p^- \textsf{ \}}_{p^- \in P_a^-} \textsf{ is a } p_1.$$

   is true in $\mathfrak{A}$ but not in $\mathfrak{B}$.

3. Similar to the previous case.

Thus if $\mathfrak{A} \not\sim_{\text{Rel}} \mathfrak{B}$, there exist Cop+Rel-formulae true in $\mathfrak{A}$ and false in $\mathfrak{B}$, and so the theorem holds. □

The insistence in clauses (2) and (3) of Definition 5.16 that we only consider elements of A and B satisfying at least one unary predicate stems directly from the syntax of Cop+Rel. Every quantified noun phrase in Cop+Rel must contain at least one (head) noun – the p in Every/some/no p – which translates to a positive occurrence of a unary predicate p in the semantics. Cop+Rel is thus unable to refer to "propertyless" entities. We can of course (as English itself indeed does)

assume a universal property satisfied by every entity, and so simplify Definition 5.16. Suppose that in every structure $\mathfrak{A}$, a special noun thing is interpreted so that $thing^{\mathfrak{A}} = \mathfrak{A}$, and consider only signatures $\Sigma = (C, P)$ such that $thing \in P$.

Corollary 5.18. *Let $\mathfrak{A}$ and $\mathfrak{B}$ be structures interpreting a signature $\Sigma = (C, P)$ such that $thing^{\mathfrak{A}} = A$ and $thing^{\mathfrak{B}} = B$. Then $\mathfrak{A}$ and $\mathfrak{B}$ have the same Cop+Rel theory if and only if*

1. *for every constant $c \in C$, $tp^{\mathfrak{A}}[c^{\mathfrak{A}}] = tp^{\mathfrak{B}}[c^{\mathfrak{B}}]$,*
2. *for every $a \in A$, there exists $b \in B$ such that $tp^{\mathfrak{A}}[a] = tp^{\mathfrak{B}}[b]$,*
3. *for every $b \in B$, there exists $a \in A$ such that $tp^{\mathfrak{A}}[a] = tp^{\mathfrak{B}}[b]$,*

Proof. Near-identical to the proof of Theorem 5.17 □

Recall from the proof of Theorem 5.5 that since Cop and Cop+Rel are both Horn fragments, we were forced to extend both of them with sentence coordination in order to obtain any kind of Invariance Theorem. As Cop+Rel is not Horn, it is natural to ask whether this requirement continues to hold, or whether an Invariance Theorem can be shown directly.

It turns out that an analogue of Theorem 5.5 also holds for certain non-Horn fragments, which we prove using the following lemma.

Lemma 5.19. *No Cop+Rel-formula, or conjunction thereof, is equivalent to $p(c) \vee q(c)$, where p, q are distinct unary predicates.*

Proof. Let $\phi_0 = p(c) \vee q(c)$ and suppose there exists some set Φ of Cop+Rel-formulae such that $\models \phi_0 \leftrightarrow \bigwedge \Phi$. There are three possibilities, when $\bigwedge \Phi$ is written with the elements of Φ in some fixed order:

1. $\bigwedge \Phi = \pm r(d) \wedge \Phi'$ for some constant d, unary predicate r and conjunction of Cop+Rel-formulae Φ'.
2. $\bigwedge \Phi = \exists x(r(x) \wedge \phi_1(x)) \wedge \Phi'$ for some unary predicate r, an N′-formula $\phi_1(x)$ and conjunction of Cop+Rel-formulae Φ'.
3. $\bigwedge \Phi = \forall x((r(x) \wedge \phi_1(x)) \rightarrow \phi_2(x)) \wedge \Phi'$ for some unary predicate r, N′-formulae $\phi_1(x), \phi_2(x)$, with $\phi_1(x)$ possibly empty, and Φ' a conjunction of Cop+Rel-formulae.

We consider each case in turn.

1. Suppose that $\bigwedge \Phi = r(d) \wedge \Phi'$ (the case $\bigwedge \Phi = \neg r(d) \wedge \Phi'$ is similar.)

   If $r \neq p$ and $r \neq q$, or if r is one of p, q and $d \neq c$, then the structure $\mathfrak{A}$ given over the domain $\{a, a'\}$ by $c^{\mathfrak{A}} = a, d^{\mathfrak{A}} = a', p^{\mathfrak{A}} = \{a\}, q^{\mathfrak{A}} = \emptyset$ and $r^{\mathfrak{A}} = \emptyset$ is a model of ϕ_0, but not a model of $r(d)$ – and hence not a model of $\bigwedge \Phi$, contradicting $\models \phi_0 \leftrightarrow \bigwedge \Phi$.

   Otherwise, $d = c$ and $r = p$ or $r = q$. Suppose without loss that $r = p$. Then $r \neq q$, since p and q are distinct. Let $\mathfrak{A}$ be the structure over the single element domain $\{a\}$ given by $c^{\mathfrak{A}} = a, p^{\mathfrak{A}} = \emptyset, q^{\mathfrak{A}} = \{a\}$. Then $\mathfrak{A}$ is a model of ϕ_0, but not of $\bigwedge \Phi$, and so again, ϕ_0 and $\bigwedge \Phi$ are not equivalent.

2. If $r = p$ or $r = q$ (suppose without loss that $r = p$), then let $\mathfrak{A}$ be the structure over the domain $\{a\}$ given by $c^{\mathfrak{A}} = a, p^{\mathfrak{A}} = \emptyset, q^{\mathfrak{A}} = \{a\}$. Then $\mathfrak{A}$ is a model of ϕ_0 and not a model of $\bigwedge \Phi$, and so $\not\models \phi_0 \leftrightarrow \bigwedge \Phi$.

   Otherwise, $p \neq r \neq q$. Then any structure $\mathfrak{A}$ such that $c^{\mathfrak{A}} \in p^{\mathfrak{A}}$ (or $c^{\mathfrak{A}} \in q^{\mathfrak{A}}$) and $r^{\mathfrak{A}} = \emptyset$ is a model of ϕ_0 and not of $\bigwedge \Phi$.

3. By the previous cases, we can see that every element of Φ must be universally quantified, of the form $\forall x((r(x) \wedge \phi_1(x)) \rightarrow \phi_2(x))$ (with $\phi_1(x)$ possibly empty). In particular, the antecedent of every such formula ϕ contains a positive occurrence $r_\phi(x)$ of a unary predicate r_ϕ. Let $\mathfrak{A}$ be any structure in which every unary predicate is assigned an empty denotation. It is immediate then that every element of Φ is true in $\mathfrak{A}$, and that ϕ_0 is false in $\mathfrak{A}$, and so $\bigwedge \Phi$ and ϕ_0 are not equivalent.

Since the preceding cases are exhaustive, it follows that no such Φ can exist. □

Theorem 5.20. *There is no relation $\sim$ on structures such that for every first-order formula ϕ, ϕ is equivalent to a Cop+Rel-formula if and only if ϕ is invariant for $\sim$.*

Proof. It is trivial to show that $\phi_0 = p(c) \vee q(c)$ is invariant for any such relation $\sim$. By the preceding lemma, however, ϕ_0 is not equivalent to any conjunction of Cop+Rel-formulae. □

The proof of Lemma 5.19 relied essentially on the structure of N′-formulae; in particular, every N′-formula contains a leading positive occurrence of a unary predicate. Since this constraint applies to the formulae generated from N′-phrases in *every* fragment we consider in this thesis, an analogue of Theorem 5.20 holds for all of them. We are therefore required to extend every fragment with rules for sentence coordination in order to obtain Invariance Theorems, which we henceforth do without comment.

So, let Cop+Rel* be the fragment of English generated by the grammar of Cop+Rel (including wh-movement), augmented with phrase structure rules for sentence coordination.

$$\mathrm{IP}/\phi \wedge \psi \to \mathrm{IP}/\phi,\ \mathsf{and},\ \mathrm{IP}/\psi$$
$$\mathrm{IP}/\phi \vee \psi \to \mathrm{IP}/\phi,\ \mathsf{or},\ \mathrm{IP}/\psi$$

Note that Cop+Rel* can contain sentences with embedded IP coordination: Every man who is a stoic { and/or } is a cynic is a philosopher. It is routine, although tedious, to show that every Cop+Rel*-sentence containing such coordination can be rewritten as an equivalent Cop+Rel*-sentence containing only coordination of complete (non-embedded) IPs. Since Cop+Rel* allows sentence negation, it follows that Cop+Rel* is simply the result of closing Cop+Rel under Boolean combinations of sentences.

With the preceding remarks in mind, the following result is obtained via a similar proof to Theorem 5.6.

Theorem 5.21. *A first-order formula ϕ is equivalent to a Cop+Rel*-formula if and only if ϕ is invariant for Rel-simulation.*

Unlike the previous two cases, the addition of sentence coordination does not affect the complexity of deciding satisfiability: the semantic complexity of Cop+Rel* is shown in Chapter 6 to be NP-complete. In fact, it is straightforward to prove that any fragment whose semantic complexity is NP-hard or harder is unaffected (in complexity terms) by the addition of sentence coordination.

5.4 Relative clauses with verbs

The obvious next set of fragments to consider are the extensions of Cop+Rel with both transitive and ditransitive verbs. We consider the various cases separately.

Each of the following fragments contain potentially scope-ambiguous sentences; however, we assume throughout that all sentences are given their surface reading.

5.4.1 Cop+Rel+TV

Let Cop+Rel+TV be the fragment of English generated by the union of the phrase structure rules of Cop+Rel with the rules TV, together with the rule of wh-movement as described in Section 5.3. An example sentence in Cop+Rel+TV is Every stoic who hates some cynic admires some philosopher, which, with the relative scope of quantifiers taken according to surface order, has the truth conditions

$$\forall x(stoic(x) \wedge \exists y(cynic(y) \wedge hate(x,y)) \rightarrow \exists z(philosopher(z) \wedge admire(x,z))).$$

Figure 5.5 shows the derivation of the preceding Cop+Rel+TV-sentence, and its semantics.

The fragment of logic generated by Cop+Rel+TV can be described in two stages.

Definition 5.22. Let a *Cop+Rel N′-formula* be an N′-formula as described in Section 5.3, pp.107. A formula $\phi(x)$ is a *Cop+Rel+TV N′-formula* if

1. $\phi(x)$ is a Cop+Rel N′-formula, or

2. $\phi(x)$ is of one of the forms $\pi(x) \wedge \pm t(x,c)$ or $\pi(x) \wedge \pm t(c,x)$, where $\pi(x)$ is a Cop+Rel N′-formula, c is a constant and t a binary predicate.

3. $\phi(x)$ is of one of the forms

   $\pi_1(x) \wedge Qy(\pi_2(y) * \tau(x,y))$
   $\pi_1(x) \wedge Qy(\pi_2(y) \wedge \psi(y) * \tau(x,y))$

   where $(Q,*) \in \{(\forall,\rightarrow),(\exists,\wedge)\}$, $\pi_1(x),\pi_2(y)$ is a Cop+Rel N′-formula, $\psi(y)$ is a Cop+Rel+TV N′-formula and $\tau(x,y)$ is a literal containing a binary predicate and the variables x,y in some order.

A formula ϕ is a Cop+Rel+TV N′-formula if it is of one of the forms $\pm p(c)$, $\pm t(c,d)$, $\forall x(\psi_1(x) \rightarrow \pm\psi_2(x))$ or $\exists x(\psi_1(x) \wedge \pm\psi_2(x))$, where c,d are constants, t is a binary predicate and $\psi_1(x),\psi_2(x)$ are Cop+Rel+TV N′-formulae, with $\psi_1(x)$ not one of the forms in 2 above.

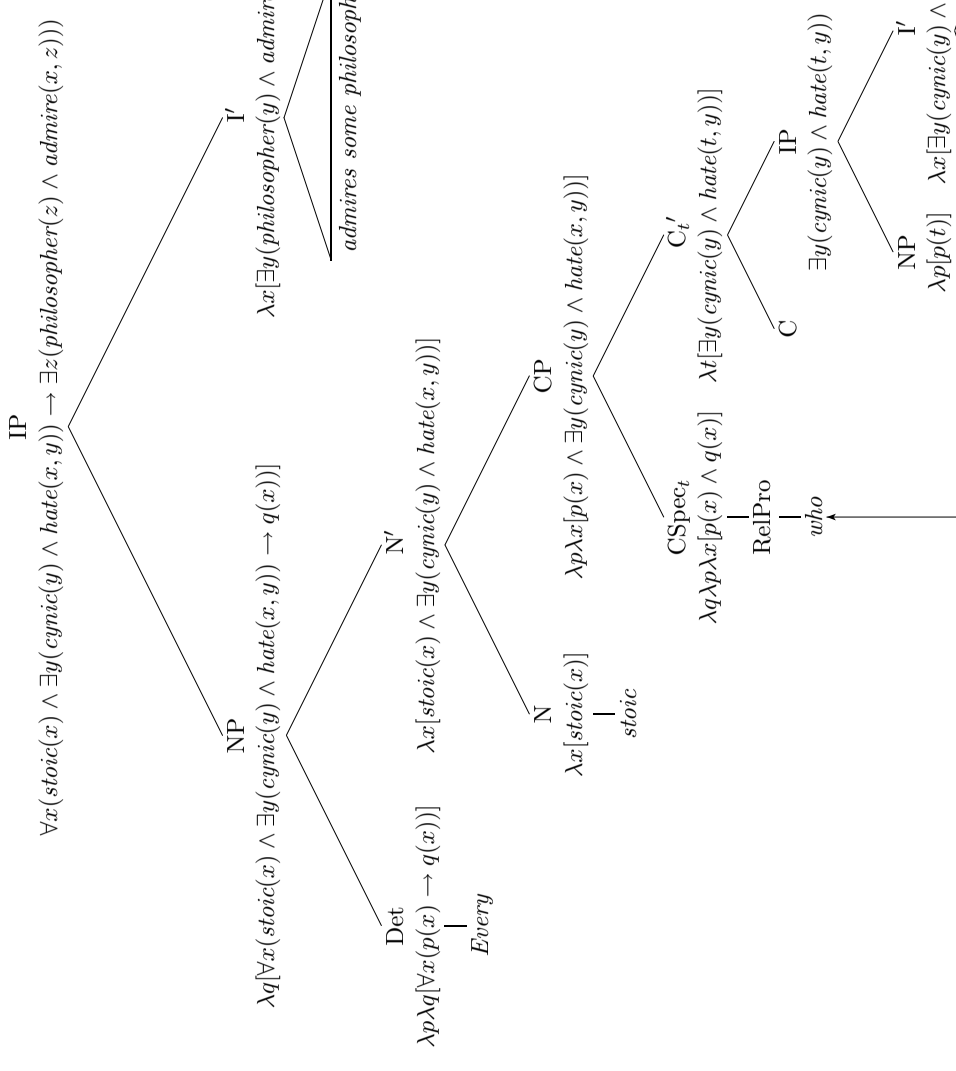


Figure 5.5: Structure of a simple Cop+Rel+TV sentence

It is hopefully clear that Cop+Rel+TV-formulae as defined here are precisely the formulae produced by generation of an IP using the phrase structure rules of Cop+Rel+TV.

Trivially, Cop+Rel+TV is at least as logically expressive as Cop, Cop+Rel and Cop+TV. Unsurprisingly, it strictly extends all of these fragments.

Proposition 5.23. *There exists structures $\mathfrak{A}$, $\mathfrak{B}$ such that $\mathfrak{A}$ and $\mathfrak{B}$ have the same Cop+Rel and Cop+TV theories (and hence the same Cop theories also), and there exists a Cop+Rel+TV sentence true in $\mathfrak{A}$ but not in $\mathfrak{B}$.*

Proof. Let $A = \{a_1, a_2, a_3, a_4\}$, let $B = \{b_1, b_2, b_3, b_4, b_5, b_6, b_7\}$, and define $\mathfrak{A}$ and $\mathfrak{B}$ by

$$\begin{array}{ll} p_1^{\mathfrak{A}} = \{a_1\} & p_1^{\mathfrak{B}} = \{b_1, b_5\} \\ p_1^{\mathfrak{A}} = \{a_2, a_3\} & p_2^{\mathfrak{B}} = \{b_2, b_3, b_6\} \\ p_3^{\mathfrak{A}} = \{a_4\} & p_3^{\mathfrak{B}} = \{b_4, b_7\} \\ t_1^{\mathfrak{A}} = \{(a_1, a_2)\} & t_1^{\mathfrak{B}} = \{(b_1, b_2), (b_5, b_6)\} \\ t_2^{\mathfrak{A}} = \{(a_3, a_4)\} & t_2^{\mathfrak{B}} = \{(b_3, b_4), (b_6, b_7)\} \end{array}$$

$\mathfrak{A}$ and $\mathfrak{B}$ are Cop+TV-similar and Cop+Rel-similar, but the Cop+Rel+TV-sentence Some p_1 t_1s some p_2 who t_2s some p_3 is true in $\mathfrak{B}$ but not in $\mathfrak{A}$. □

Semantic Complexity

The combination of relative clauses and transitive verbs has a marked effect on complexity. The following result was shown in [58].

Theorem 5.24. *The problem of determining satisfiability in Cop+Rel+TV is EXPTIME-complete.*

In order to prove one of the results in Chapter 6, (Theorem 6.15, to be precise), we adapt the EXPTIME-hardness proof for Cop+Rel+TV. It suffices to note here that every Cop+Rel+TV-sentence required for this hardness proof is of one of the following forms:

Every p which is (not) a q is an r

Every p is a q

No p is a q

Every p ts some q

Every p which ts some q is an r

Expressive Power

In preceding sections, we studied the expressive power of a fragment F firstly by defining a relation on structures corresponding to the preservation of truth of F-formulae, and then showing an Invariance Theorem for the extension of F with sentence coordination, since we know by Theorem 5.20 that no Invariance Theorem holds for F directly.

In order to obtain similar results for Cop+Rel+TV, and, indeed, all of the fragments remaining in this chapter, we need to make a small extension to the grammars we use. In particular, we need to allow the conjunction of relative clauses, so that as well as Every stoic who hates some cynic is a philosopher in Cop+Rel+TV, we also allow Every stoic who hates some cynic and whom every cynic hates is a philosopher, with the obvious corresponding semantics. Towards the end of this section, we show that such extension is relatively harmless.

Definition 5.25. Let $\mathfrak{A}$ and $\mathfrak{B}$ interpret a signature $\Sigma = (C, P, T)$ consisting of constants C, unary predicates P and binary relations T. Let A' be the subset of A such that every $a \in A'$ either satisfies some unary predicate or is the interpretation of some constant in $\mathfrak{A}$, and likewise for B'. A *Rel+TV-simulation* $R \subseteq A' \times B'$ is a relation satisfying the following conditions:

1. for every constant c, $c^{\mathfrak{A}} R c^{\mathfrak{B}}$.
2. for every $a \in A', b \in B'$ such that aRb, $\mathrm{tp}^{\mathfrak{A}}[a] = \mathrm{tp}^{\mathfrak{B}}[b]$,
3. for every $a \in A', b \in B'$ such that aRb, and every constant symbol c, $\mathrm{tp}^{\mathfrak{A}}[a, c^{\mathfrak{A}}] = \mathrm{tp}^{\mathfrak{B}}[b, c^{\mathfrak{B}}]$,
4. for every $a \in A', b \in B'$ such that aRb, and every $t \in T$,
   (a) if, for some $a' \in A'$, $\mathfrak{A} \models \pm t[a, a']$, then for some $b' \in B'$ such that $a'Rb'$, $\mathfrak{B} \models \pm t[b, b']$,
   (b) if, for some $a' \in A'$, $\mathfrak{A} \models \pm t[a', a]$, then for some $b' \in B'$ such that $a'Rb'$, $\mathfrak{B} \models \pm t[b', b]$,
   (c) if, for some $b' \in B'$, $\mathfrak{B} \models \pm t[b, b']$, then for some $a' \in A'$ such that $a'Rb'$, $\mathfrak{A} \models \pm t[a, a']$,
   (d) if, for some $b' \in B'$, $\mathfrak{B} \models \pm t[b', b]$, then for some $a' \in A'$ such that $a'Rb'$, $\mathfrak{A} \models \pm t[a', a]$,

5. for every $a \in A'$, there exists $b \in B'$ such that aRb,

6. for every $b \in B'$, there exists $a \in A'$ such that aRb,

$\mathfrak{A}$ and $\mathfrak{B}$ are *Rel+TV-similar*, written $\mathfrak{A} \sim_{\text{Rel+TV}} \mathfrak{B}$, if some $R \subseteq A' \times B'$ is a Rel+TV-simulation.

Lemma 5.26. *If $\mathfrak{A} \sim_{Rel+TV} \mathfrak{B}$, then $\mathfrak{A}$ and $\mathfrak{B}$ agree on all Cop+Rel+TV formulae.*

Proof. Let $R \subseteq A' \times B'$ be a Rel+TV simulation.

By 1 and 3, $\mathfrak{A}$ and $\mathfrak{B}$ agree on all ground Cop+Rel+TV formulae.

A simple structural induction shows that for all N′-formulae $\phi(x)$, if $a \in A', b \in B'$ such that aRb, then $\mathfrak{A} \models \phi[a]$ iff $\mathfrak{B} \models \phi[b]$.

Now let ϕ be any non-ground Cop+Rel+TV formula, and suppose that $\mathfrak{A} \models \phi$. In each case, $\mathfrak{B} \models \phi$. We give only one example here, for brevity – the rest are similar. Let $\psi_1(x)$ and $\psi_2(x)$ be N′-formulae.

$\phi = \forall x(\psi_1(x) \to \psi_2(x))$: For all $a \in A$, if $\mathfrak{A} \models \psi_1[a]$, then $\mathfrak{A} \models \psi_2[a]$. Suppose for some $b \in B$, $\mathfrak{B} \models \psi_1[b] \wedge \neg\psi_2[b]$. If $b \in B'$, then by 6, there exists $a \in A'$ such that aRb and, by the above induction, $\mathfrak{A} \models \psi_1[a] \wedge \neg\psi_2[a]$ – a contradiction. So $b \in B \setminus B'$. By definition, $\psi_1(x)$ contains a unary predicate p such that every element which satisfies $\psi_1(x)$ in any structure satisfies $p(x)$, contradicting the assumption that $b \in B \setminus B'$. So $\mathfrak{B} \models \phi$.

□

Lemma 5.27. *Let $\mathfrak{A}$ and $\mathfrak{B}$ be ω-saturated structures which agree on all formulae in Cop+Rel+TV. Then $\mathfrak{A} \sim_{Rel+TV} \mathfrak{B}$.*

Proof. Define $R \subseteq A' \times B'$ as follows:

$$R = \{(a, b) \mid a \in A', b \in B', \text{ and for all N′-formulae } \phi(x), \\ \mathfrak{A} \models \phi[a] \text{ iff } \mathfrak{B} \models \phi[b]\}$$

We check that R is a Rel+TV-simulation.

1. Immediate, since $\mathfrak{A}$ and $\mathfrak{B}$ have the same Cop+Rel+TV theory.

2. Any 1-type can be specified fully by an N′-formula in the fragment Cop+Rel. We use the assumption that a and b both satisfy at least one predicate in their respective structures here.

3. Let $a \in A'$, let c be a constant symbol and let p be a unary predicate symbol such that $\mathfrak{A} \models p(a)$. For every binary predicate t such that $\mathfrak{A} \models \pm t[a, c^{\mathfrak{A}}]$, $p(x) \wedge \pm t(x, c)$ is an N′-formula satisfied by a in $\mathfrak{A}$, and hence for all $b \in B$ such that aRb, $\mathfrak{B} \models p[b] \wedge \pm t[b, c^{\mathfrak{B}}]$. A similar argument applies in the case $\mathfrak{A} \models \pm t[c^{\mathfrak{A}}, a]$. It then follows by 1 and 2 that $\mathrm{tp}^{\mathfrak{A}}[a, c^{\mathfrak{A}}] = \mathrm{tp}^{\mathfrak{B}}[b, c^{\mathfrak{B}}]$.

4. (We consider only case 4a) Suppose that for some $a \in A'$, there exists a binary t such that $\mathfrak{A} \models \pm t[a, a']$ for some $a' \in A'$. We show that for every $b \in B'$ such that aRb, there exists $b' \in B'$ such that $\mathfrak{B} \models \pm t[b, b']$ and $a'Rb'$. Let Φ be the set of N′-formulae satisfied by a'. For every finite sequence $\phi_1(x), \ldots, \phi_n(x)$ of elements of Φ, a satisfies the N′-formula $\psi(x) = \exists y(\phi_1(y) \wedge \ldots \wedge \phi_n(y) \wedge \pm t(x, y))$ in $\mathfrak{A}$, and so, for all $b \in B'$ such that aRb, $\mathfrak{B} \models \psi[b]$. $\psi(x)$ is an N′-formula only because Cop+Rel+TV allows conjunction of relative clauses. By ω-saturation of $\mathfrak{B}$, there exists $b' \in B'$ such that $\mathfrak{B} \models \bigwedge_{\phi(y) \in \Phi} \phi[b'] \wedge \pm t[b, b']$. That is, b' satisfies precisely the N′-formulae satisfied by a', and hence $a'Rb'$.

5. Let $a \in A'$, let p be a unary predicate satisfied by a in $\mathfrak{A}$ and let Φ be the set of N′-formulae satisfied by a in $\mathfrak{A}$. Then for every finite sequence of elements $\phi_1(x), \ldots, \phi_n(x)$ of Φ, $\mathfrak{A} \models p[a] \wedge \phi_1[a] \wedge \ldots \wedge \phi_n[a]$, and hence $\mathfrak{A} \models \exists x(p(x) \wedge \phi_1(x) \wedge \ldots \wedge \phi_n(x))$ – a Cop+Rel+TV formula (since we allow conjunction of relative clauses), which must therefore be true in $\mathfrak{B}$ also. So for each finite sequence of elements $\phi_1(x), \ldots, \phi_n(x)$ of Φ, there exists $b \in B'$ such that $\mathfrak{B} \models p[b] \wedge \phi_1[b] \wedge \ldots \wedge \phi_n[b]$, and so by ω-saturation of $\mathfrak{B}$, there exists $b \in B'$ such that $\mathfrak{B} \models p[b] \wedge \bigwedge_{\phi(x) \in \Phi} \phi[b]$. Since Φ is closed under negation, it follows that aRb.

6. Similar to 5.

Thus R is a Rel+TV-simulation, and so $\mathfrak{A} \sim_{\mathrm{Rel+TV}} \mathfrak{B}$. □

Let Cop+Rel+TV* be the extension of Cop+Rel+TV with coordination of complete sentences. In light of Lemmas 5.26 and 5.27, we can now prove

Theorem 5.28. *A first-order formula ϕ is equivalent to a Cop+Rel+TV* formula if and only if ϕ is invariant for Rel+TV-simulation.*

Proof. Essentially the same as the proof of Theorem 5.6, save that, having built structures $\mathfrak{A}$ and $\mathfrak{B}$ with the same Cop+Rel+TV* theory, we cannot immediately

conclude that $\mathfrak{A} \sim_{\text{Rel+TV}} \mathfrak{B}$. However, as every structure has an ω-saturated elementary extension (by Theorem 3.16), we are able to construct elementary extensions $\mathfrak{A}^*$ and $\mathfrak{B}^*$ of $\mathfrak{A}$ and $\mathfrak{B}$ respectively such that $\mathfrak{A}^* \sim_{\text{Rel}} \mathfrak{B}^*$, whence the rest of the proof proceeds as earlier. □

5.4.2 Cop+Rel+DTV

Let Cop+Rel+DTV be the fragment of English generated by the union of the phrase structure rules of Cop+Rel with the rules DTV, together with the rule of wh-movement as described in Chapter 4, Section 4.4.1, in much the same way as Cop+Rel+TV was formed from Cop+Rel. An example sentence in Cop+Rel+DTV is Every philosopher who is a stoic prefers every stoic to every cynic, with the semantics

$$\forall x(\mathit{philosopher}(x) \wedge \mathit{stoic}(x) \rightarrow \forall y(\mathit{stoic}(y) \rightarrow \forall z(\mathit{cynic}(z) \rightarrow \mathit{prefer}(x,y,z))))$$

The remainder of this section proceeds along very similar lines to the preceding section.

Definition 5.29. A formula $\phi(x)$ is a *Cop+Rel+DTV N′-formula* if

1. $\phi(x)$ is a Cop+Rel N′-formula, or
2. $\phi(x)$ is of one of the forms
   $\pi_1(x) \wedge Qy(\pi_2(y) * \delta(x,y,c))$
   $\pi_1(x) \wedge Qy(\pi_2(y) \wedge \psi(y) * \delta(x,y,c))$
   $\pi_1(x) \wedge \delta(x,c,d)$
   where $(Q,*) \in \{(\forall,\rightarrow),(\exists,\wedge)\}$, c,d are constants, $\pi_1(x),\pi_2(y)$ are Cop+Rel N'-formulae, $\psi(y)$ is a Cop+Rel+DTV N'-formula and $\delta(t_1,t_2,t_3)$ is a literal containing a ternary predicate and the terms t_1,t_2,t_3 in some order.
3. $\phi(x)$ is of one of the forms
   $\pi(x) \wedge Q_1y(\pi_1(y) *_1 Q_2z(\pi_2(z) \wedge \psi_2(z) *_2 \delta(x,y,z)))$
   $\pi(x) \wedge Q_1y(\pi_1(y) \wedge \psi_1(y) *_1 Q_2z(\pi_2(z) *_2 \delta(x,y,z)))$
   $\pi(x) \wedge Q_1y(\pi_1(y) *_1 Q_2z(\pi_2(z) \wedge \psi_2(z) *_2 \delta(x,y,z)))$
   $\pi(x) \wedge Q_1y(\pi_1(y) \wedge \psi_1(y) *_1 Q_2z(\pi_2(z) \wedge \psi_2(z) *_2 \delta(x,y,z)))$
   where $(Q_1,*_1),(Q_2,*_2) \in \{(\forall,\rightarrow),(\exists,\wedge)\}$, $\pi(x),\pi_1(y),\pi_2(z)$ are Cop+Rel N'-formulae, $\psi_1(y),\psi_2(z)$ are Cop+Rel+DTV N'-formulae and $\delta(t_1,t_2,t_3)$

is a literal containing a ternary predicate and the terms t_1, t_2, t_3 in some order.

A formula ϕ is a Cop+Rel+DTV N′-formula if it is of one of the forms $\pm p(c)$, $\pm d(c, d, e)$, $\forall x(\psi_1(x) \rightarrow \pm\psi_2(x))$ or $\exists x(\psi_1(x) \wedge \pm\psi_2(x))$, where c, d, e are constants, d is a ternary predicate and $\psi_1(x), \psi_2(x)$ are Cop+Rel+DTV N′-formulae, with $\psi_1(x)$ not one of the forms in 2 above.

It should come as no surprise that Cop+Rel+DTV strictly extends Cop+Rel, nor that it has different expressive power to both Cop+TV and Cop+Rel+TV – that is, there are structures which can be distinguished by Cop+Rel+DTV but not by Cop(+Rel)+TV, and vice versa. The generation of relevant examples is routine.

Semantic Complexity

In [60], it was shown that the obvious extension of Cop+Rel+DTV with transitive verbs, Cop+Rel+TV+DTV (to which we return later) has a NEXPTIME-complete satisfiability problem. Since Cop+Rel+DTV is by definition a subfragment of Cop+Rel+TV+DTV, its satisfiability problem is certainly also decidable in NEXPTIME.

The NEXPTIME-hardness part of the proof in [60] involves the reduction of a known NEXPTIME-hard satisfiability problem to an equisatisfiable set of sentences in Cop+Rel+TV+DTV. The set of sentences required are all of one of the following forms.

Every p_1 which is a p_2 ds every p_3 which is a p_2 to every p_4
Every p_1 which is (not) a p_2 is a p_3
Every p_1 ds some p_2 to every p_3
No p_1 ds any p_2 to every p_3
Every p_1 ds every p_2 to every p_3
No p_1 is a p_2
Some p_1 is a p_1

Since none of those sentences involve any transitive verbs, they are all in fact sentences in Cop+Rel+DTV, and so we can conclude:

Theorem 5.30. *The satisfiability problem for Cop+Rel+DTV is NEXPTIME-complete.*

Expressive Power

We make the same extension to Cop+Rel+DTV as we did to Cop+Rel+TV, and allow the conjunction of nominal phrases. Since both fragments have very similar sentence structure, it should come as no surprise that the required definitions are also similar.

Definition 5.31. Let $\mathfrak{A}$ and $\mathfrak{B}$ interpret a signature $\Sigma = (C, P, D)$ consisting of constants C, unary predicates P and ternary relations D. Let A' be the subset of A such that every $a \in A'$ either satisfies some unary predicate or is the interpretation of some constant in $\mathfrak{A}$, and likewise for B'. A *Rel+DTV-simulation* $R \subseteq A' \times B'$ is a relation satisfying the following conditions:

1. for every constant c, $c^{\mathfrak{A}} R c^{\mathfrak{B}}$.
2. for all $a \in A', b \in B'$ such that aRb, $\mathrm{tp}^{\mathfrak{A}}[a] = \mathrm{tp}^{\mathfrak{B}}[b]$,
3. for all $a \in A', b \in B'$ such that aRb, and every pair of constants $c, c' \in C$, $\mathrm{tp}^{\mathfrak{A}}[a, c^{\mathfrak{A}}, c'^{\mathfrak{A}}] = \mathrm{tp}^{\mathfrak{B}}[b, c^{\mathfrak{B}}, c'^{\mathfrak{B}}]$,
4. for all $a \in A', b \in B'$ such that aRb, every $c \in C$ and every $d \in D$,
   (a) if, for some $a' \in A'$, $\mathfrak{A} \models d(a, a', c^{\mathfrak{A}})$, then for some $b' \in B'$ such that $a'Rb'$, $\mathfrak{B} \models d(b, b', c^{\mathfrak{B}})$, and similarly for every permutation of the order of the arguments of d.
   (b) if, for some $b' \in B'$, $\mathfrak{B} \models d(b, b', c^{\mathfrak{B}})$, then for some $a' \in A'$ such that $a'Rb'$, $\mathfrak{A} \models d(a, a', c^{\mathfrak{A}})$, and similarly for every permutation of the order of the arguments of d.
5. for all $a_1 \in A', b_1 \in B'$ such that $a_1 R b_1$, and every $d \in D$,
   (a) if, for some $a_2, a_3 \in A'$, $\mathfrak{A} \models \pm d[a_1, a_2, a_3]$, then for some $b_2, b_3 \in B'$ such that $a_2 R b_2, a_3 R b_3$, $\mathfrak{B} \models \pm d[b_1, b_2, b_3]$, and similarly for every permutation of the order of the arguments of d.
   (b) if, for some $b_2, b_3 \in B'$, $\mathfrak{B} \models \pm d[b_1, b_2, b_3]$, then for some $a_2, a_3 \in A'$ such that $a_2 R b_2, a_3 R b_3$, $\mathfrak{A} \models \pm d[a_1, a_2, a_3]$, and similarly for every permutation of the order of the arguments of d.
6. for all $a \in A'$, there exists $b \in B'$ such that aRb,

7. for all $b \in B'$, there exists $a \in A'$ such that aRb,

$\mathfrak{A}$ and $\mathfrak{B}$ are *Rel+DTV-similar*, written $\mathfrak{A} \sim_{\text{Rel+DTV}} \mathfrak{B}$, if some $R \subseteq A' \times B'$ is a Rel+DTV-simulation.

Lemma 5.32. *If $\mathfrak{A} \sim_{Rel+DTV} \mathfrak{B}$, then $\mathfrak{A}$ and $\mathfrak{B}$ agree on all Cop+Rel+DTV-formulae.*

Proof. Essentially the same as the proof of Lemma 5.26. □

Lemma 5.33. *Let $\mathfrak{A}$ and $\mathfrak{B}$ be ω-saturated structures which agree on all formulae in Cop+Rel+DTV. Then $\mathfrak{A} \sim_{Rel+DTV} \mathfrak{B}$.*

Proof. Define $R \subseteq A' \times B'$ as follows:

$$R = \{(a, b) \mid a \in A', b \in B', \text{ and for all N}'\text{-formulae } \phi(x), \\ \mathfrak{A} \models \phi[a] \text{ iff } \mathfrak{B} \models \phi[b]\}$$

We show that R is a Rel+DTV-simulation. Most of the clauses in Definition 5.31 can be shown to hold of R via similar arguments to those employed in the proof of Lemma 5.27. The only slightly more complicated case is 4. We prove part 4a here – 4b then follows by symmetry of $\mathfrak{A}$ and $\mathfrak{B}$.

Let $a_1, a_2 \in A', c \in C$ such that $\mathfrak{A} \models \delta[a_1, a_2, c^{\mathfrak{A}}]$. By 6, there are $b_1, b_2 \in B'$ such that a_1Rb_1, a_2Rb_2 and so by definition of R, a_1 and b_1 satisfy the same N′-formulae, as do a_2 and b_2. In particular, let $\pi_1(x), \pi_2(y)$ be Cop+Rel N′-formulae fixing the 1-types of a_1, a_2, respectively. Then $\pi_1(x) \wedge \exists y(\pi_2(y) \wedge \delta(x, y, c))$ and $\pi_2(x) \wedge \exists y(\pi_1(y) \wedge \delta(x, y, c))$ are N′-formulae satisfied by a_1 and a_2, respectively, and so are satisfied by b_1 and b_2, respectively, in $\mathfrak{B}$. It follows that $\mathfrak{B} \models \delta[b_1, b_2, c^{\mathfrak{B}}]$, as required, and the result holds. □

By the now-familiar argument, we can show:

Theorem 5.34. *Let Cop+Rel+DTV* be the fragment Cop+Rel+DTV extended with coordination of complete sentences. Then a first-order formula ϕ is equivalent to a formula in Cop+Rel+DTV* if and only if ϕ is invariant for Rel+DTV-simulation.*

5.4.3 Cop+Rel+TV+DTV

As alluded to in the previous section, Cop+Rel+TV+DTV is the fragment of English generated by the union of the grammar of Cop+Rel+TV with the rules DTV. So, for example, the sentence Every stoic who knows some epicurean prefers every epicurean to every cynic is in Cop+Rel+TV+DTV, and is assigned the semantics

$$\forall x(\mathit{stoic}(x) \wedge \exists y(\mathit{epicurean}(y) \wedge \mathit{know}(x,y)) \rightarrow \\ \forall y(\mathit{epicurean}(y) \rightarrow \forall z(\mathit{cynic}(z) \rightarrow \mathit{prefer}(x,y,z)))).$$

The fragment of logic generated from Cop+Rel+TV+DTV is that generated by the obvious combination of Definitions 5.22 and 5.29 in a similar manner to the definitions of both Cop+Rel+TV and Cop+Rel+DTV.

It is trivial to show that Cop+Rel+TV+DTV offers a strict increase in expressive power over all fragments considered so far.

Semantic Complexity

In [60], the following result was proved.

Theorem 5.35. *Satisfiability in Cop+Rel+TV+DTV is NEXPTIME-complete.*

Expressive Power

Owing to the structure of Definitions 5.25 and 5.31, expressive power results for Cop+Rel+TV+DTV are surprisingly straightforward. First, we add conjunction of relative clauses to Cop+Rel+TV+DTV, and then proceed as before.

Definition 5.36. Let $\Sigma = (C, P, T, D)$ be a signature consisting of constants C, unary predicates P, binary relations T and ternary relations D. Let $\mathfrak{A}, \mathfrak{B}$ be structures interpreting Σ, let A' be the subset of A such that every $a \in A'$ either satisfies some unary predicate or is the interpretation of some constant in $\mathfrak{A}$, and likewise for B', and let $R \subseteq A' \times B'$. We say that R is a Rel+TV+DTV-simulation if

1. R is a Rel+TV-simulation.
2. R is a Rel+DTV-simulation.

Structures $\mathfrak{A}$ and $\mathfrak{B}$ are Rel+TV+DTV-similar, written $\mathfrak{A} \sim_{\text{TV+DTV}} \mathfrak{B}$, if some $R \subseteq A' \times B'$ is a Rel+TV+DTV-simulation.

Lemma 5.37. *If $\mathfrak{A} \sim_{TV+DTV} \mathfrak{B}$, then $\mathfrak{A}$ and $\mathfrak{B}$ agree on all Cop+Rel+TV-DTV-formulae.*

Proof. Essentially the same as the proofs of Lemmas 5.26 and 5.32. □

Lemma 5.38. *Let $\mathfrak{A}$ and $\mathfrak{B}$ be ω-saturated structures which agree on all formulae in Cop+Rel+TV+DTV. Then $\mathfrak{A} \sim_{TV+DTV} \mathfrak{B}$.*

Proof. Essentially the same as the proofs of Lemmas 5.27 and 5.33. □

We can therefore conclude

Theorem 5.39. *Let Cop+Rel+TV+DTV* be the fragment Cop+Rel+TV+DTV extended with coordination of complete sentences. Then a first-order formula ϕ is equivalent to a formula in Cop+Rel+TV+DTV* if and only if ϕ is invariant for Rel+TV+DTV-simulation.*

Of course, all of the preceding results, from Lemma 5.26 onwards, depend on the assumption that arbitrary finite conjunction of relative clauses is allowed. It might be hoped, of course, either that some version of these results which did not rely on such an assumption, or a result similar to Theorem 5.20 showing its necessity, might be found. Currently, the question remains open.

However, in a certain sense, the assumption of arbitrary relative clause conjunction can be seen as logically harmless, as it is possible, using only sentences in Cop+Rel+TV(+DTV), to "define out" the coordinations necessary for the proofs of Lemmas 5.26 and 5.27 (or Lemmas 5.32 and 5.33, or Lemmas 5.37 and 5.38), provided we allow extensions to the relevant signature Σ. For every N′-formula $\phi(x)$ over Σ corresponding to a relative clause, let p_ϕ be a *fresh* unary predicate symbol, and let $\psi_1^\phi = \forall x(p_\phi(x) \rightarrow \phi(x))$ and $\psi_2^\phi = \forall x(\phi(x) \rightarrow p_\phi(x))$. Let Q be the set of p_ϕ for all $\phi(x)$.

For every $\phi(x)$, both ψ_1^ϕ and ψ_2^ϕ are Cop+Rel+TV(+DTV)-formulae over Σ extended with Q.

Now, let $\phi_1(x)$ and $\phi_2(x)$ be any N′-formulae over Σ corresponding to relative clauses, and let ϕ be any formula in Cop+Rel+TV(+DTV) over Σ containing $\phi_1(x) \wedge \phi_2(x)$ as a subformula. Then $p_{\phi_1}, p_{\phi_2} \in Q$. Let ϕ' be the result of replacing every occurrence of $\phi_1(x) \wedge \phi_2(x)$ in ϕ with $p_{\phi_1}(x) \wedge p_{\phi_2}(x)$.

Clearly, $\phi', \psi_1^{\phi_1}, \psi_1^{\phi_2}, \psi_2^{\phi_1}, \psi_2^{\phi_2}$ together entail ϕ, and any structure $\mathfrak{A}$ in which ϕ holds can be extended to a structure $\mathfrak{A}'$ interpreting both Σ and Q such that $\mathfrak{A}$ and $\mathfrak{A}'$ have the same Cop+Rel+TV(+DTV) theories over Σ and such that $\mathfrak{A}' \models \phi' \wedge \psi_1^{\phi_1} \wedge \psi_1^{\phi_2} \wedge \psi_2^{\phi_1} \wedge \psi_2^{\phi_2}$. By repeating this process as necessary, we can ensure that every Cop+Rel+TV(+DTV) formula involving the conjunction of N′-formula needed in the proofs of the relevant lemmas can be replaced with Cop+Rel+TV(+DTV)-formulae over $\Sigma \cup Q$ *not* involving the conjunction of N′-formulae, without harming the outcome of the proof.

Every new formula used for this "defining out" process is clearly a member of Cop+Rel+TV(+DTV), so it does not seem too far-fetched to claim that we have not exceeded the expressive-power bounds of Cop+Rel+TV(+DTV) in proving the foregoing results.

5.5 Conclusion

In this chapter, we have investigated both the semantic complexity and the expressive power of a range of fragments of English constructed from simple, everyday grammatical constructions. The main contributions are the purely-semantic characterisations of each fragment given in Theorems 5.6, 5.12, 5.21, 5.28, 5.34, 5.39, the semantic complexity result for Cop+TV(+DTV) in Theorem 5.8 and the method given in Section 5.2 for the automatic generation of a truth-preserving relation on structures for arbitrary finite fragments satisfying certain properties. Table 5.5 summarises the semantic complexity results. Figure 5.5 summarises the comparative expressive powers of the fragments studied. A solid line between two fragments indicates that the lower fragment is strictly more expressive than the fragment above it, and a dashed line indicates that two fragments have simply different expressive powers, with neither capable of distinguishing all pairs of situations distinguishable by the other.

In the following chapter, we proceed to extend each of the fragments considered in this chapter with various levels of subsentential coordination and prove semantic complexity results in each case.

Fragment	Complexity
Cop	PTIME
Cop+TV	PTIME
Cop+TV+DTV	PTIME
Cop+Rel	NP-complete
Cop+Rel+TV	EXPTIME-complete
Cop+Rel+DTV	NEXPTIME-complete
Cop+Rel+TV+DTV	NEXPTIME-complete

Table 5.1: Summary of semantic complexity results

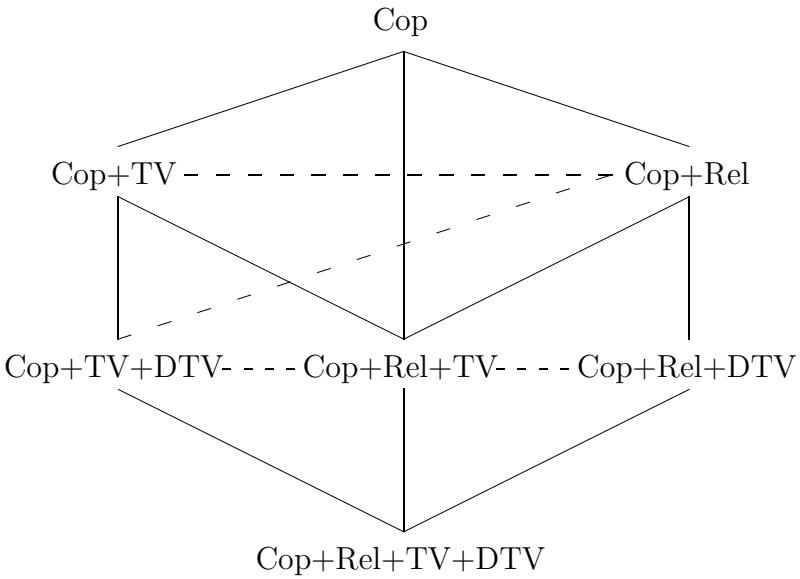


Figure 5.6: Comparative expressive powers

Chapter 6

Coordination

You may call it combination, you may call it the accidental and fortuitous concurrence of atoms.

— Lord Palmerston, on a projected Palmerston-Disraeli coalition, 5 March 1857

In this chapter, we take each of the fragments considered in the preceding chapter, and extend their grammars with rules for a variety of forms of coordination. That is to say, we separately extend each fragment in turn with rules for coordination of full sentences, noun phrases, nominal phrases, relative clauses, predicate phrases and verbs. For each fragment so generated, we analyse its semantic complexity. Questions of the expressive power of each of these fragments remain open.

6.1 Sentences

We have already mentioned the fragments obtained by extending the fragments of Chapter 5 with rules for the coordination of sentences. Each of the starred fragments used in the Invariance Theorems in the previous chapter is of this kind.

Let sentence coordination be defined by the phrase structure rules

$$\mathrm{IP}/\phi \wedge \psi \rightarrow \mathrm{IP}/\phi,\mathsf{and},\mathrm{IP}/\psi$$
$$\mathrm{IP}/\phi \vee \psi \rightarrow \mathrm{IP}/\phi,\mathsf{or},\mathrm{IP}/\psi$$

subject to the restriction that no coordinated IP contains an unbound wh-trace. The reason for this restriction is simply to rule out examples such as *Every man who admires some stoic and some cynic despises. We return to the coordination of relative clauses later, in Section 6.4.

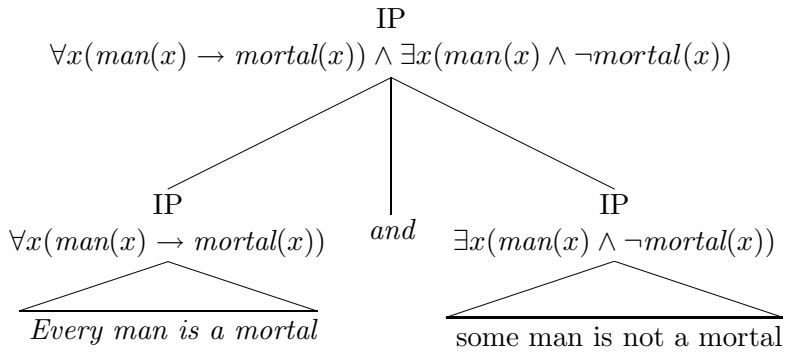


Figure 6.1: Structure of a simple Cop*-sentence

Now let F be any decidable fragment of English, of those defined in the previous chapter, which does *not* contain sentence coordination, and let F* be the extension of F with the above rules. For example, Figure 6.1 shows the derivation of a sentence and its semantics in Cop∗.

We give a decision procedure for F*.

Theorem 6.1. *Let F and F* be as above. Then the satisfiability problem for F* is decidable.*

Proof. Let E be any set of F*-sentences and let Φ be the set of translations of E into first-order logic. We can, by definition of F*, write Φ in the form

$$(\phi_{1,1} \vee \ldots \vee \phi_{1,n_1}) \wedge (\phi_{2,1} \vee \ldots \vee \phi_{2,n_2}) \wedge \ldots \wedge (\phi_{m,1} \vee \ldots \vee \phi_{m,n_m})$$

where for all $1 \leq i \leq m$, $1 \leq j \leq n_i$, $\phi_{i,j}$ is an F-formula.

To determine the satisfiability of Φ, simply guess an assignment of truth values to each $\phi_{i,j}$ such that for all $1 \leq i \leq m$, there is at least one j, $1 \leq j \leq n_i$ such that $\phi_{i,j}$ is assigned "true". We can write this assignment as a set of F-formulae $\Psi = \{\pm\phi_{1,1}, \ldots, \pm\phi_{m,n_m}\}$ such that for all $1 \leq i \leq m$, $1 \leq j \leq n_i$, $\phi_{i,j} \in \Psi$ if $\phi_{i,j}$ is assigned "true", and $\neg\phi_{i,j} \in \Psi$ otherwise.

Since F is decidable, we can decide the satisfiability of Ψ. If it is satisfiable, then Φ is satisfiable. Otherwise, try again.

The above procedure forms a non-deterministic algorithm for deciding the satisfiability of E. □

The non-determinism of the above algorithm only has a visible effect on the complexity of satisfiability for the smallest among our fragments.

Theorem 6.2. *Let F be one of Cop, Cop+TV or Cop+TV+DTV, and let F^* be the result of extending F with sentence coordination. Then the satisfiability problem for F^* is NP-complete.*

Proof. Membership in NP is easily shown. In the notation of Theorem 6.1, guessing the assignment leading to the construction of Ψ takes no more than polynomial time, and for each of the listed fragments, the satisfiability of Ψ can be checked in polynomial time.

To show NP-hardness, we reduce propositional satisfiability to satisfiability in Cop*, thus establishing the result for all of the listed fragments. It is well known that it is only necessary to consider a set Φ of propositional clauses all of the forms $P \vee Q$, $\neg P \vee \neg Q$ or $\neg P \vee \neg Q \vee R$, where P, Q, R are proposition letters.

Let c be a constant and for each proposition letter P, let p be a unary predicate symbol. Let Ψ be the result of replacing every occurrence of every proposition letter P in Φ with the atom $p(c)$. Thus Ψ consists only of clauses of the forms $p(c) \vee q(c)$, $\neg p(c) \vee \neg q(c)$ or $\neg p(c) \vee \neg q(c) \vee r(c)$, and every element of Ψ is a Cop*-formula. It is straightforward to show that Φ and Ψ are equisatisfiable, and so satisfiability in Cop* is NP-hard. □

Theorem 6.3. *Let F be any of the fragments of Chapter 5 containing relative clauses, and let F^* be the result of extending the grammar of F with sentence coordination. Then the satisfiability problem for F^* has the same complexity as the satisfiability problem for F.*

Proof. The satisfiability problem for Cop+Rel is NP-complete, and so the nondeterminism of the algorithm in the proof of Theorem 6.1 makes no difference to complexity.

All of the remaining fragments have a satisfiability problem which is at best in EXPTIME. In the notation of the proof of Theorem 6.1, all the possibilities for the construction of Ψ can be attempted in deterministic exponential time, and so again, complexity is unaffected. □

6.2 Noun Phrases

We now turn to forms of subsentential coordination, beginning with that of noun phrases.

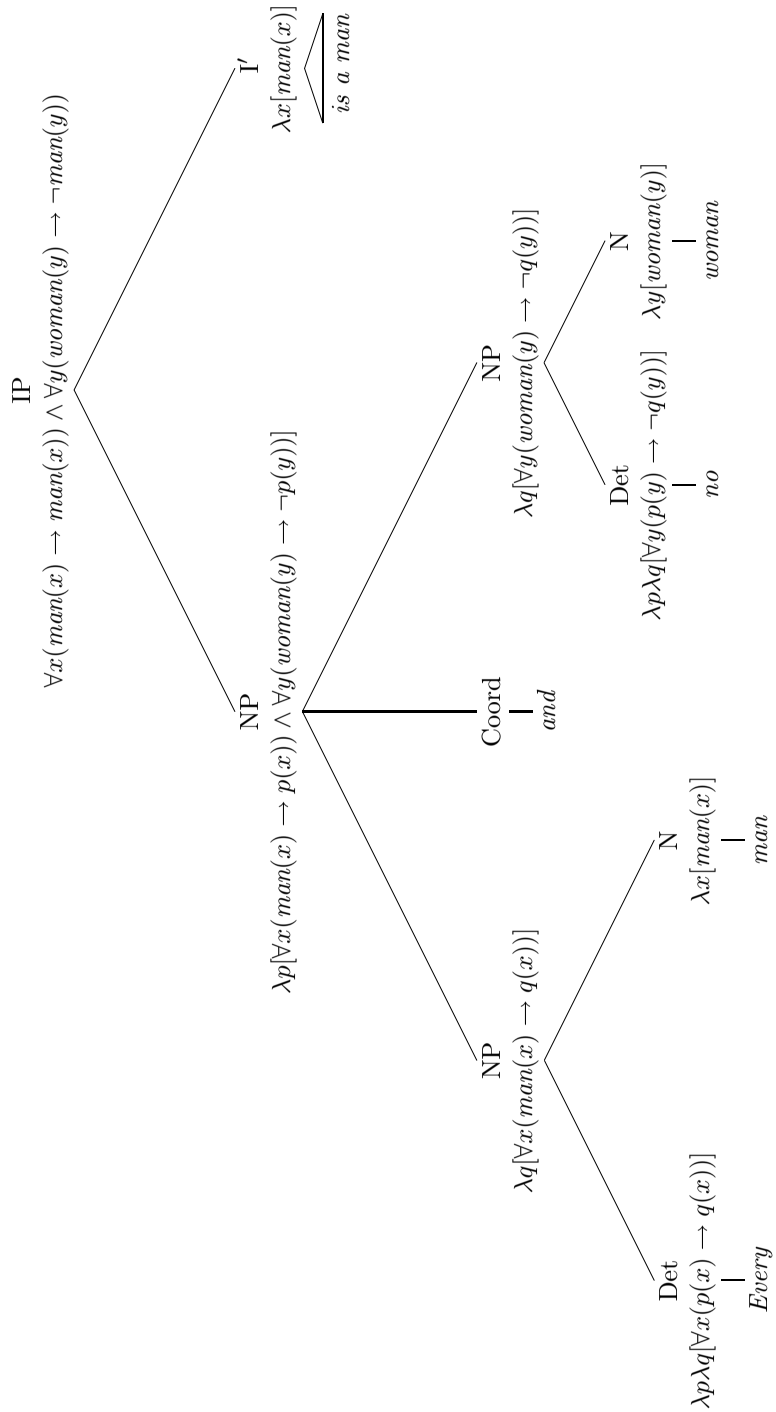


Figure 6.2: Structure of a simple Cop+NPCoord-sentence

Let F be one of the fragments of Chapter 5. We let F+NPCoord be the fragment obtained by extending the grammar of F with the following rules.

$$\mathrm{NP}/\lambda p[\phi(p) \wedge \psi(p)] \rightarrow \mathrm{NP}/\phi,\mathsf{and},\mathrm{NP}/\psi$$
$$\mathrm{NP}/\lambda p[\phi(p) \vee \psi(p)] \rightarrow \mathrm{NP}/\phi,\mathsf{or},\mathrm{NP}/\psi$$

subject to the restriction that no NPs containing an unbound wh-trace can be coordinated.

It is easily verified that these rules give the appropriate semantics to coordinations of NPs. Figure 6.2 shows an example in Cop+NPCoord.

For the sake of simplicity, we ignore issues of number agreement for coordinated NPs. In any example sentences in which it matters, we silently replace ungrammatical constructions such as *Socrates and Plato is a mortal with Socrates and Plato are mortals without comment. We also exclude from the content lexicon any relational nouns, in order to rule out sentences such as Tom, Dick and Harry are brothers – while grammatical, such sentences have quite different semantics and do not properly belong in any of the fragments we consider here. We do not trouble ourselves either to handle the elided coordinator between, for example, Tom and Dick in the preceding example. The semantics are identical whether we perform the elision or not.

Each F+NPCoord generates a fragment of logic in the usual way. We may therefore ask what the semantic complexity of each fragment is.

Theorem 6.4. *The satisfiability problem for Cop+NPCoord is NP-complete.*

Proof. Membership in NP follows immediately from the observation that coordination of *subject* NPs, which is all that can occur in Cop+NPCoord, generates semantics identical to those of a coordination of sentences in Cop. The computation of semantics in Figure 6.2 demonstrates why this claim is true; every such computation must proceed in a similar manner. Theorem 6.2 then gives membership in NP.

To show NP-hardness, we reduce a special case of propositional satisfiability to satisfiability in Cop+NPCoord. We can restrict attention to propositional clauses of the forms $\neg P \vee \neg Q \vee \neg R$ or $P \vee Q$, where P, Q, R are proposition letters, without affecting NP-hardness (see, e.g., [46]). Let Φ be any set of such clauses, and for every proposition letter P occurring in Φ, let p be a proper noun. Let t be a common noun, and let E_ϕ be the set of Cop+NPCoord-sentences produced by translating the elements of Φ as follows.

$\neg P \vee \neg Q \vee \neg R$	p or q or r is not a t	$\neg t(p) \vee \neg t(q) \vee \neg t(r)$
$P \vee Q$	p or q is a t	$t(p) \vee t(q)$

Let Ψ be the set of translations of elements of E_ϕ into first-order logic. Then Φ and Ψ differ in size by only a constant factor and are visibly equisatisfiable. The NP-hardness, and thus NP-completeness, of the satisfiability problem for Cop+NPCoord follows. □

Unfortunately, the precise semantic complexities of Cop+TV+NPCoord and Cop+TV+DTV+NPCoord are not yet known, although we can give the following partial results.

Theorem 6.5. *The satisfiability problem for Cop+TV+NPCoord is at best NP-hard, and is decidable in EXPTIME.*

Proof. NP-hardness is immediate, since every Cop+NPCoord-formula is also in Cop+TV+NPCoord.

To see membership in EXPTIME, we make several observations. Note first that, as above, all coordination of subject NPs in Cop+TV+NPCoord generate sentences in Cop+TV*.

Sentences containing conjunctions of object NPs are also equisatisfiable with sentences in Cop+TV*. Consider any sentence S of the form NP$_1$ *t*s NP$_2$ and NP$_3$. If NP$_1$ is a proper noun, or is *not* specified by the determiner some, then S is equivalent to NP$_1$ *t*s NP$_2$ and NP$_1$ *t*s NP$_3$. Otherwise, NP$_1$ is of the form some N', and S is equisatisfiable with c *t*s NP$_2$ and c *t*s NP$_3$, where c is a fresh common noun. (In terms of logic, we can Skolemise the interpretation of NP$_1$.)

As for disjunctions of object NPs, every sentence of the form NP$_1$ *t*s NP$_2$ or NP$_3$ is equisatisfiable with a sentence containing relative clauses of the form NP$_1$ who does not t NP$_2$ *t*s NP$_3$ or NP$_1$ who does not t NP$_3$ *t*s NP$_2$ if NP$_1$ is not a proper noun, and with a sentence of the form NP$_1$ *t*s NP$_2$ or NP$_1$ *t*s NP$_3$ otherwise.

Of course, when applied to NP disjunctions with more than two disjuncts, this rewriting could lead to sentences such as

Every man [who does not admire Tom]
[who does not admire Dick]
admires Harry

with both relative clauses qualifying man – a construction we explicitly noted was ungrammatical in Chapter 4, Section 4.4.1. However, by introducing fresh unary

predicates as in Chapter 5, Section 5.4, we can always eliminate such sentences without affecting the problem size significantly.

Thus every Cop+TV+NPCoord-sentence can be rewritten in polynomial time as an equisatisfiable sentence in Cop+Rel+TV*. Since satisfiability in the latter fragment is in EXPTIME, so is satisfiability in the former. □

Theorem 6.6. *The satisfiability problem for Cop+TV+DTV+NPCoord is at best NP-hard, and is decidable in NEXPTIME.*

Proof. Very similar to the proof of Theorem 6.5. □

For the remaining fragments, the addition of NP coordination has no effect on semantic complexity.

Theorem 6.7. *The satisfiability problem for Cop+Rel+NPCoord is NP-complete.*

Proof. Observe that every NP in a Cop+Rel+NPCoord-sentence occurs in subject position, and so just as for the earlier cases, each Cop+Rel+NPCoord-sentence can be rewritten as a sentence in Cop+Rel*. NP-completeness follows from Theorems 5.15 and 6.3. □

Theorem 6.8. *Cop+Rel+TV+NPCoord has an EXPTIME-complete satisfiability problem, whereas the satisfiability problems for Cop+Rel+DTV+NPCoord and Cop+Rel+TV+DTV+NPCoord are both NEXPTIME-complete.*

Proof. By introducing new unary predicates where necessary, sentences in any of the listed fragments can be rewritten so that NP coordinations do not occur within relative clauses. But then the rewriting methods of the proof of Theorem 6.5 apply, leading to equisatisfiable sentences containing no NP coordination, but featuring relative clauses and sentence coordination. Each of the above fragments already contains relative clauses, and the addition of sentence coordination has no effect on semantic complexity, as shown in Theorem 6.3. The result then follows from Theorems 5.24, 5.30 and 5.35. □

6.3 Nominals

Let us move within the noun phrase now, and consider what happens when we allow the coordination of nominal phrases. That is, what is the logical effect of allowing phrases such as every man or woman and some scholar and gentleman?

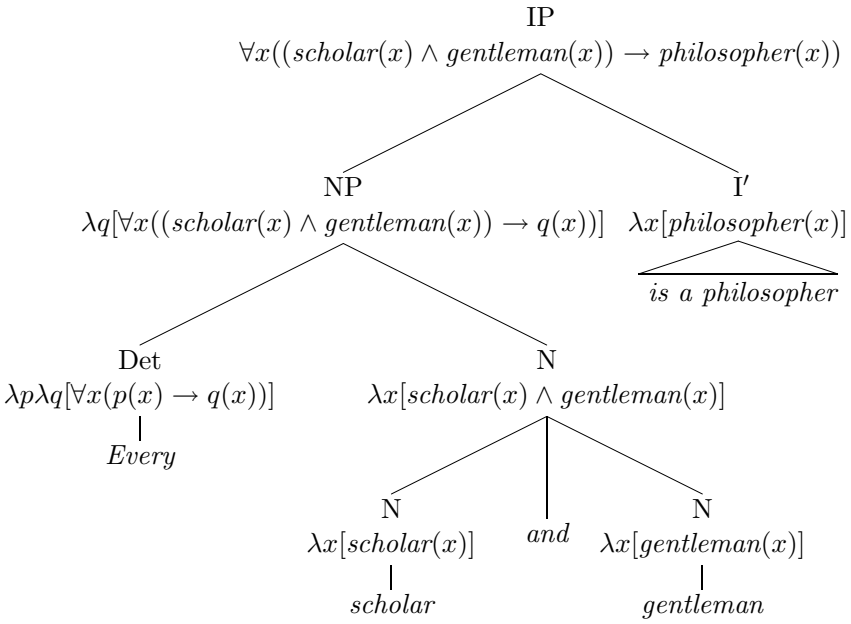


Figure 6.3: Structure of a simple Cop+N′Coord-sentence

We need to take some care with the choice of examples here. We wish to restrict attention to cases where, *semantically*, the coordination takes place between the denotations of nominals. So some scholar and gentleman refers to an entity which is both a scholar and a gentleman. Different choices of common noun rule out this kind of reading, for external reasons: every man and woman is usually read as every man and every woman rather than every thing which is both a man and a woman, and is semantically closer to NP coordination. We take it that this interpretation is forced by world knowledge regarding men and women rather than by any syntactic property. So we must take care to select example phrases and sentences which do allow the desired reading.

The following annotated phrase structure rules suffice to generate the correct phrases when added to any of our list of fragments.

$$\mathrm{N}'/\lambda x[\phi(x) \wedge \psi(x)] \rightarrow \mathrm{N}'/\phi,\mathsf{and},\mathrm{N}'/\psi$$
$$\mathrm{N}'/\lambda x[\phi(x) \vee \psi(x)] \rightarrow \mathrm{N}'/\phi,\mathsf{or},\mathrm{N}'/\psi$$

If F is a fragment of English, let F+N′Coord be the fragment generated by extension of the grammar of F with the above rules. Figure 6.3 shows the structure of a sentence in Cop+N′Coord, along with its generated semantics.

We can show upper bounds for the semantic complexity of all fragments containing N′ coordination at once: the proof does not depend in the particular

properties of any fragment. So, let F+N′Coord be as above, and let F+Rel be F extended with relative clauses, if it does not already contain them.

Theorem 6.9. *For every fragment F in Chapter 5, the satisfiability problem for F+N′Coord reduces to the satisfiability problem for F+Rel.*

Proof. Let F be any of the relevant fragments, and let Φ be an arbitrary set of F+N′Coord-formulae. If $\phi \in \Phi$ is the semantic interpretation of a sentence containing N′ coordination, then ϕ must contain a subformula $\phi(x)$ of the form $\phi_1(x) \vee \phi_2(x)$ or $\phi_1(x) \wedge \phi_2(x)$, where $\phi_1(x), \phi_2(x)$ are either of the same form as $\phi(x)$, or are N′-formulae in the fragment F. We rewrite each such ϕ in the usual way, as follows.

Suppose that ϕ contains a subformula $\phi(x)$ of the form $\phi_1(x) \vee \phi_2(x)$. Let p be a fresh unary predicate, let ψ be the result of replacing every occurrence of $\phi(x)$ in ϕ with $p(x)$ and let Ψ_p be the defining clause $\forall x(p(x) \wedge \neg\phi_1(x) \rightarrow \phi_2(x)$. Then the set of formulae $\Psi = (\Phi \setminus \{\phi\}) \cup \{\psi, \Psi_p\}$ is equisatisfiable with Φ. A similar rewriting can be performed for all $\phi(x) = \phi_1(x) \wedge \phi_2(x)$ occurring in Φ. We can repeat this process until we obtain an equisatisfiable set Ψ^* only polynomially larger than Φ. By construction, every formula in Ψ^* is an F+Rel-formula, and the result follows. □

For each fragment already containing relative clauses, Theorem 6.9 also gives a lower bound.

Theorem 6.10. *The satisfiability problem for*

1. *Cop+Rel+N′Coord is NP-complete.*
2. *Cop+Rel+TV+N′Coord is EXPTIME-complete.*
3. *Cop+Rel+DTV+N′Coord is NEXPTIME-complete.*
4. *Cop+Rel+TV+DTV+N′Coord is NEXPTIME-complete.*

Proof. Immediate from Theorem 6.9 and the relevant theorems in Chapter 5. □

It turns out that extending Cop with N′ coordination also yields the same lower complexity bound as the corresponding extension with relative clauses.

Theorem 6.11. *The satisfiability problem for Cop+N′Coord is NP-complete.*

Proof. In [58], satisfiability in Cop+Rel is shown to be NP-hard by the reduction of propositional satisfiability to the satisfiability of sentences in Cop (which are also therefore in Cop+N′Coord) and Cop+Rel-sentences of the forms

Every p which is not a q is an r
Every p which is a q is an r

where p, q and r are common nouns. But these sentences are easily shown to be equivalent to Cop+N′Coord-sentences of the forms

Every p is an r or q
Every p and q is an r

respectively. Thus the satisfiability problem for Cop+N′Coord is NP-hard, and thus NP-complete. □

For the remaining fragments Cop+TV and Cop+TV+DTV, the question of a lower complexity bound remains open.

6.4 Relative Clauses

Given that the determiners every, some and no cannot be coordinated with each other – *every and some man makes little sense – the only other element of the noun phrase which can be coordinated (among the fragments we consider, at least) is the relative clause. In this section, then, we study the effects of allowing phrases such as every stoic who admires some cynic or whom some platonist despises.

In order to allow wh-movement to operate correctly, we interpret such phrases as being generated by the coordination of CPs, as in

$[_{NP}$ every stoic $[_{CP}$ $[_{CP}$ who$_i$ t_i admires every cynic] and
$[_{CP}$ whom$_j$ some platonist despises t_j]]]

The other possible position for relative clause coordination is at the level of the embedded IP, but the data seem to argue against this option:

$[_{NP}$ *every stoic $[_{CP}$ who$_i$ $[_{IP}$ $[_{IP}$ t_i admires every cynic] and
$[_{IP}$ some platonist despises t_i]]]]

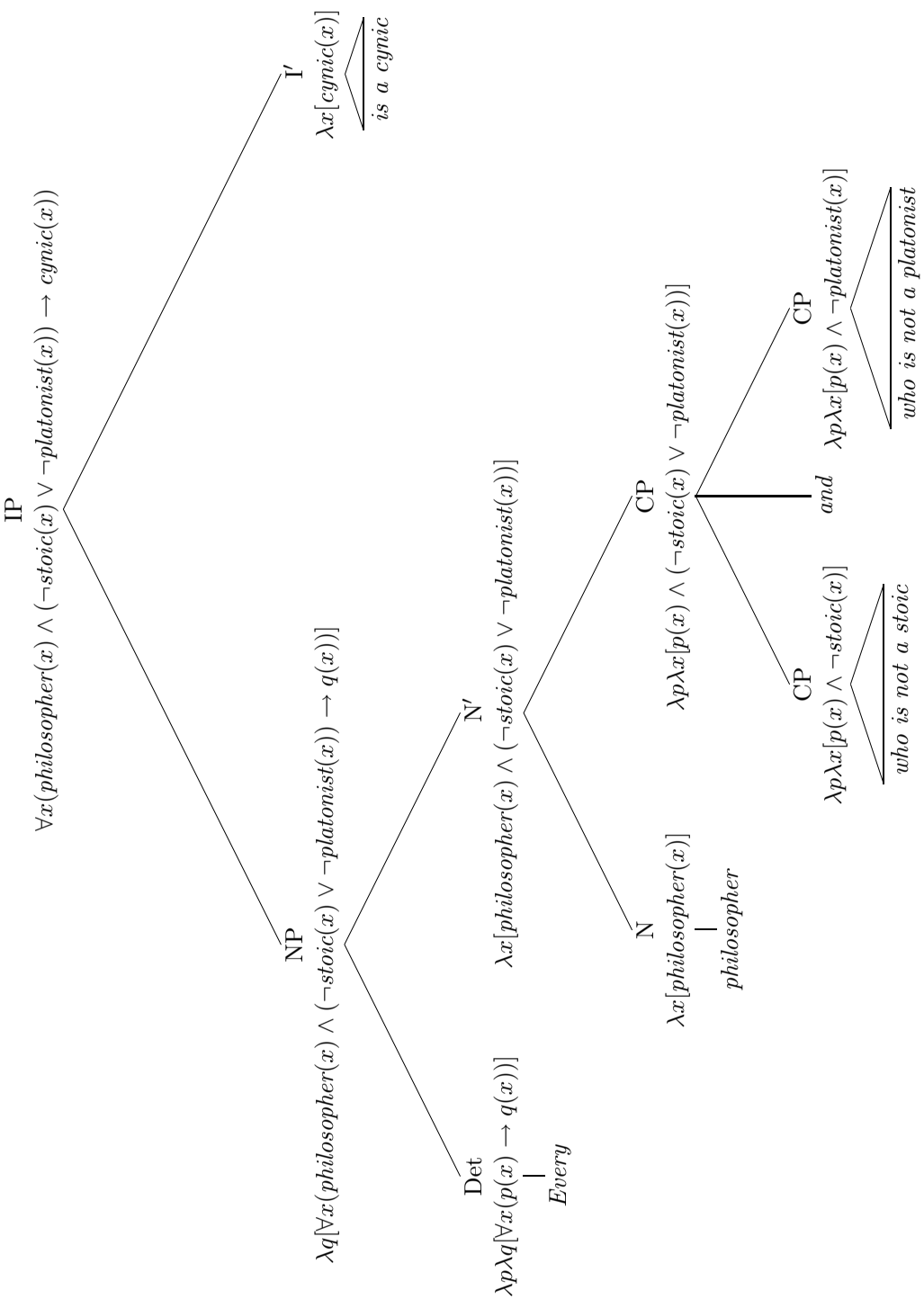


Figure 6.4: Structure of a simple Cop+Rel+CPCoord sentence

One possible reason for the ungrammaticality of this example is that, since it occurs in both subject and object positions, the trace t_i cannot be assigned a unique case.

We therefore generate coordinations of relative clauses using the following rules

$$\mathrm{CP}/\lambda p\lambda x[(\phi(p))[x] \wedge (\psi(p))[x]] \rightarrow \mathrm{CP}/\phi,\mathsf{and},\mathrm{CP}/\psi$$
$$\mathrm{CP}/\lambda p\lambda x[(\phi(p))[x] \vee (\psi(p))[x]] \rightarrow \mathrm{CP}/\phi,\mathsf{or},\mathrm{CP}/\psi$$

In the usual way, for each fragment F, let F+CPCoord be the fragment generated by adding these rules to the grammar of F. Figure 6.4 shows the structure of a sentence containing relative clause coordination, and the semantics generated for it.

Of course, it only makes sense to add relative clause coordination to fragments which already contain relative clauses. We thus only look at the appropriate extensions of Cop+Rel, Cop+Rel+TV, Cop+Rel+DTV and Cop+Rel+TV+DTV. Given all the results so far in this chapter, it ought not to come as much of a surprise that semantic complexity is unaffected in any of these cases.

Theorem 6.12. *Let F be one of the fragments listed above. Then the semantic complexity of F+CPCoord is the same as the semantic complexity of F.*

Proof. By construction of each F, every occurrence of CP coordination is in a phrase of the form

$$\begin{array}{lll}[_{N'}\ p\ [_{CP} & [_{CP}\ \mathsf{who(m)}\ \ldots] & \mathsf{and/or} \\ & [_{CP}\ \mathsf{who(m)}\ \ldots]]] & \end{array}$$

where p is some common noun. But clearly, every such phrase can be equivalently replaced with a phrase

$$\begin{array}{lll}[_{N'} & [_{N'}\ p\ [_{CP}\ \mathsf{who(m)}\ \ldots]\ ]\ & \mathsf{and/or} \\ & [_{N'}\ p\ [_{CP}\ \mathsf{who(m)}\ \ldots]]] & \end{array}$$

since the semantics of each coordinate in both of these are forced to have the same single free variable, which is therefore bound by a single quantifier. The reduction used in the proof of Theorem 6.9 applies, so that each F+CPCoord is a subfragment of F+Rel, which is the same as F for all of the cases listed here. □

6.5 Predicate phrases

The preceding sections have exhausted the possibilities for coordination within the noun phrase. We now move on to consider coordination between other elements of the sentence: the copula, verb phrases and, within the VP, the verb itself.

In all of the fragments of English we have studied, the only methods of predication of a subject are via the copula, or a verb phrase (in those fragments which contain verbs.) Rather than dealing with coordination of these constituents separately, we consider them together, under the banner of "predicate coordination".

Both the copula and the verb phrase, and their negations, are immediately dominated by the same category in our grammars: they are both daughters of I′, which in each case takes its semantics unmodified from its Cop or VP daughter. We may thus handle both Cop and VP coordination in a uniform way, as I′ coordination. The following rules suffice.

$$\mathrm{I}'/\lambda x[\phi(x) \wedge \psi(x)] \rightarrow \mathrm{I}'/\phi,\mathsf{and},\mathrm{I}'/\psi$$
$$\mathrm{I}'/\lambda x[\phi(x) \vee \psi(x)] \rightarrow \mathrm{I}'/\phi,\mathsf{or},\mathrm{I}'/\psi$$

I′ coordinations can occur within relative clauses, as in

Every stoic who [$_{I'}$ [$_{I'}$ admires some cynic] and
[$_{I'}$ despises some platonist]].

To handle examples such as these, we need to modify our account of wh-movement slightly. Let us allow more than one relative pronoun to be moved into the same CSpec position, *provided* that no two such relative pronouns originate within the same IP, and that each begins with the same role, either subject or object, within its initial IP. In order to satisfy the Coordinate Structure Constraint, we must also restrict I′ coordination so that if any one coordinate contains an unbound wh-trace, then so do all of its sister coordinates. Figure 6.5 shows the structure of a sentence containing I′ coordination, including the modified wh-movement.

If F is any of the fragments of Chapter 5, let F+I′Coord be the result of extending the grammar of F with the above rules.

Theorem 6.13. *Let F be any one of the fragments of Chapter 5. The satisfiability problem for F+I′Coord is reducible to the satisfiability problem for F+Rel*.*

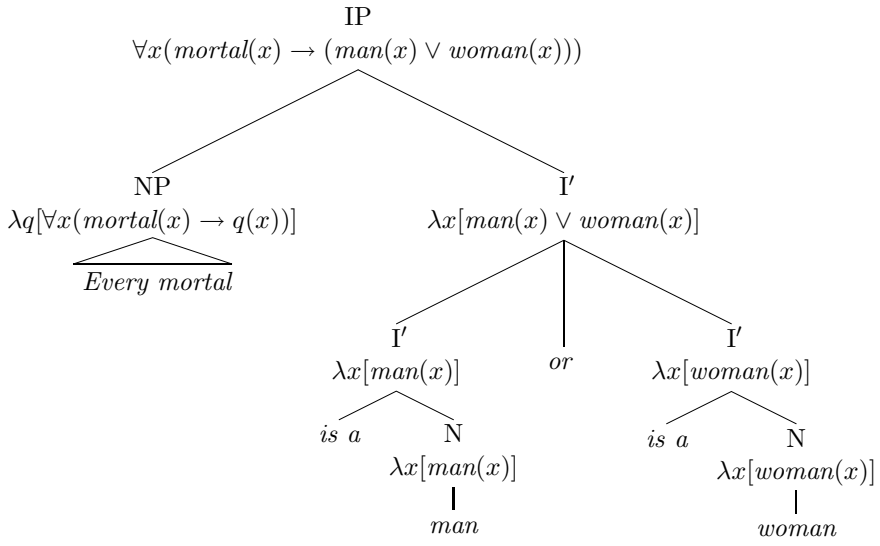


Figure 6.5: Structure of a simple Cop+I′Coord-sentence

Proof. We show only the case where I′ coordinations are *not* embedded within a relative clause: the embedded case is trivially reducible to CP coordination, which as we noted in the proof of Theorem 6.12 is reducible to ordinary relative clauses. We also assume, for both ease of presentation and linguistic nicety, that no "mixed" coordinations occur – that is, we do not encounter sentences such as ?Socrates is a man or is a mortal and admires some cynic. Nothing essential changes in the following proof, however, if such sentences are admitted.

So, let F be any fragment, and suppose we have an F+I′Coord sentence S meeting the aforementioned conditions and containing I′ coordination. S then consists of a subject NP s and an I′ coordination I'_1 and/or ... and/or I'_n. If s is a proper noun, then S is visibly equivalent to the F* sentence

$$s\ \mathsf{I}'_1\ \text{and/or}\ \ldots\ \text{and/or}\ s\ \mathsf{I}'_n.$$

Otherwise s is an NP headed by a common noun. If the main coordinator is and, then S is equivalent to the sentence

$$s\ \text{who}\ \mathsf{I}'_1\ \ldots\ \text{who}\ \mathsf{I}'_{n-1}\ \mathsf{I}'_n.$$

If the main coordinator is or, S is equivalent to the sentence

$$s\ \text{who}\ \neg\mathsf{I}'_1\ \ldots\ \text{who}\ \neg\mathsf{I}'_{n-1}\ \mathsf{I}'_n$$

where for $1 \leq i \leq n$, $\neg \mathsf{I}'_i$ is the negation of I'_i according to the grammar of F.

As it stands, these generated sentences are not F+Rel-sentences, since they may contain more than one relative clause qualifying the same noun. However, they can be converted into equisatisfiable sets of F+Rel-sentences by introducing fresh common nouns and defining-sentences of F in the usual way. Thus every F+I′Coord-sentence can be rewritten as an equisatisfiable F+Rel*-sentence. Theorems 6.2 and 6.3 then give upper complexity bounds for each fragment F+I′Coord. □

For each fragment containing relative clauses, a lower complexity bound is immediate from Theorem 6.3. We consider the remaining cases in turn.

Theorem 6.14. *The satisfiability problem for Cop+I′Coord is NP-complete.*

Proof. By the preceding theorem, satisfiability in Cop+I′Coord is decidable in NP. The NP-hardness proof is very similar to the proof of Theorem 6.11: the sentence-forms which we there rewrote in Cop+N′Coord, we must rewrite here in Cop+I′Coord. The equivalences given below suffice.

$$\begin{array}{lcl} \text{Every } p \text{ which is not a } q \text{ is an } r & \leftrightarrow & \text{Every } p \text{ is a } q \text{ or is an } r \\ \text{Every } p \text{ which is a } q \text{ is an } r & \leftrightarrow & \text{Every } p \text{ is an } r \text{ or is not a } q \end{array}$$

where p, q and r are common nouns. NP-completeness is then immediate. □

Theorem 6.15. *The satisfiability problem for Cop+TV+I′Coord is EXPTIME-complete.*

Proof. Membership in EXPTIME follows from Theorem 6.13.

In Chapter 5, Section 5.4, we listed the sentence-forms needed to prove the EXPTIME-hardness of Cop+Rel+TV. These consist of sentences in Cop+TV (which are also therefore in Cop+TV+I′Coord), Cop+Rel-sentences of the forms considered in the proof of the previous theorem (which can therefore be expressed in Cop+TV+I′Coord) and Cop+Rel+TV-sentences of the form

$$\text{Every } p \text{ which } t\text{s some } q \text{ is an } r$$

where p, q, r are common nouns and t is a transitive verb. Any such sentence can be rewritten as the equivalent Cop+TV+I′Coord sentence

$$\text{Every } p \text{ is an } r \text{ or does not } t \text{ some } q$$

from which EXPTIME-completeness follows. □

Theorem 6.16. *Cop+TV+DTV+I′Coord has a NEXPTIME-complete satisfiability problem.*

Proof. In Chapter 5, Section 5.4, we cited the NEXPTIME-hardness proof for Cop+Rel(+TV)+DTV from [60], and gave a list of all Cop+Rel+DTV-sentences required for that result. Rather than repeat them all here, we simply note that the only sentence-form needed which is not already in Cop+TV+DTV+I′Coord, or, by the proofs of the previous two theorems, is not already known to be expressible in Cop+TV+DTV+I′Coord, is

Every p which is a q ds every r which is a q to every s

where p, q, r and s are common nouns and d is a ditransitive verb.

By introducing a fresh common noun t, we can replace all such sentences with the equisatisfiable set of sentences

Every p ds every t to every s or is not a q
Every r is a t or is not a q

and hence derive NEXPTIME-completeness. □

6.6 Verbs

The final form of coordination we consider is that of verbs, both transitive and ditransitive. Is the semantic complexity of any of our fragments affected by the admission of sentences such as Plato admires and respects Socrates?

We are, of course, restricted to consideration only of those fragments which already contain verbs of some kind. Thus throughout this section, when we speak of "a fragment F", we mean any fragment of Chapter 5 except Cop and Cop+Rel.

The phrase structure rules which generate coordinations of verbs, and the relevant semantic annotations, are straightforward.

$$\mathrm{TV}/\lambda s\lambda x[(\phi(s))[x] \wedge (\psi(s))[x]] \rightarrow \mathrm{TV}/\phi,\mathsf{and},\mathrm{TV}/\psi$$
$$\mathrm{TV}/\lambda s\lambda x[(\phi(s))[x] \vee (\psi(s))[x]] \rightarrow \mathrm{TV}/\phi,\mathsf{or},\mathrm{TV}/\psi$$

$$\mathrm{DTV}/\lambda s\lambda t\lambda x[([\phi(s)](t))[x] \wedge ([\psi(s)](t))[x]] \rightarrow \mathrm{DTV}/\phi,\mathsf{and},\mathrm{DTV}/\psi$$
$$\mathrm{DTV}/\lambda s\lambda t\lambda x[(\phi(s))[x] \vee (\psi(s))[x]] \rightarrow \mathrm{DTV}/\phi,\mathsf{or},\mathrm{DTV}/\psi$$

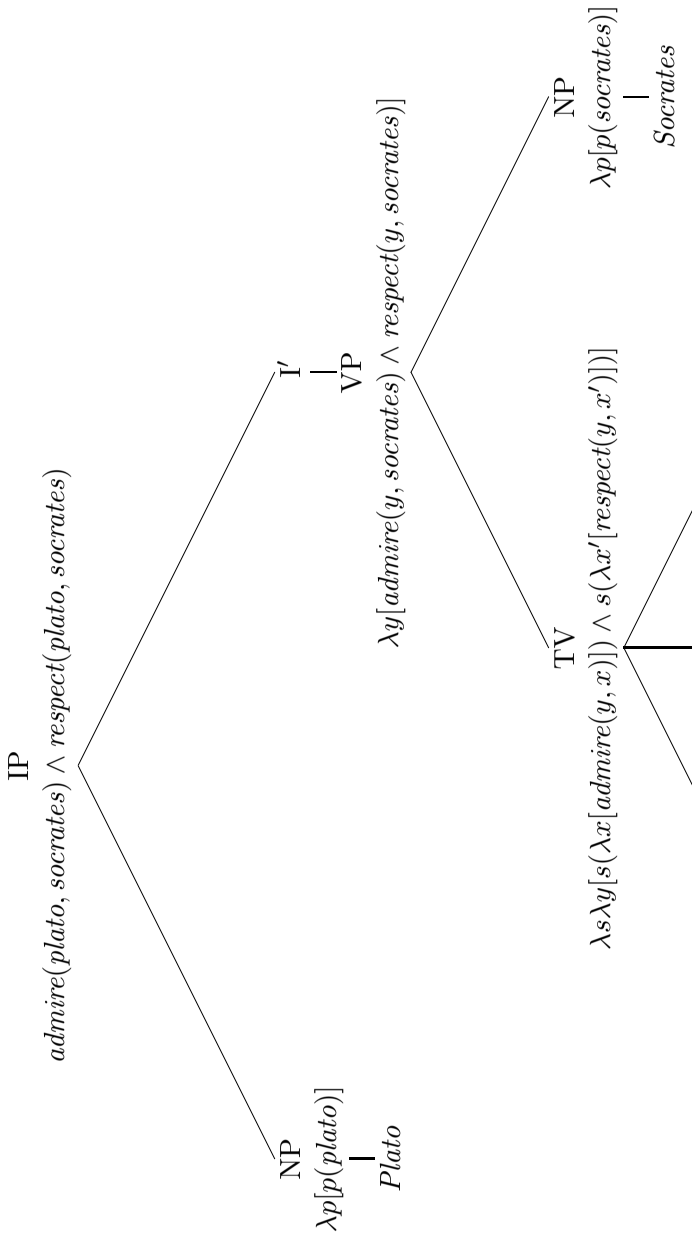


Figure 6.6: Structure of a simple Cop+TV+VCoord-sentence

We only allow like-with-like coordination – transitive verb with transitive verb, ditransitive with ditransitive. It may be possible to generalise slightly to allow "right-node raised" constructions, such as Every girl admired, and gave a flower to, some policeman, by treating the combination of a ditransitive verb, its direct object and the preposition to, as a transitive verb. However, transitive verbs can be simulated using ditransitive verbs and a "dummy" direct object proper noun: simply replace sentences of the form NP_1 ts NP_2 with NP_1 ts c to $\mathsf{NP}_b 2$, where t is a transitive verb (or a coordination of transitive verbs), t' is ditransitive (or a coordination of ditransitive verbs) depending on t and c is a proper noun. If the same noun c is used in the rewriting of every sentence containing transitive verbs, then satisfiability is clearly unharmed. Thus no higher semantic complexity can result from generalising to "mixed" coordinations. For the sake, then, of keeping the grammars simple, we stick to the basic cases.

For any fragment F, let F+VCoord be the result of extending the grammar of F with the above rules. Figure 6.6 shows the structure and semantics of a sentence containing verb coordination.

We determine the semantic complexity in each case.

Theorem 6.17. *The satisfiability problem for Cop+TV+VCoord is NP-complete.*

Proof. To show membership in NP, let E be a set of Cop+TV+VCoord-sentences, and let Φ be the result of translating E into first-order logic. By the introduction of fresh binary predicates where necessary, Φ can be converted in polynomial time into an equisatisfiable set $\Phi' = \Phi_1 \cup \Phi_2$, where Φ_1 is a set of Cop+TV-formulae, and Φ_2 contains only formulae of the forms

$$\forall x \forall y (t_1(x, y) \rightarrow t_2(x, y))$$
$$\forall x \forall y (t_1(x, y) \rightarrow t_2(x, y) \vee t_3(x, y))$$

where t_1, t_2 and t_3 are binary relations.

The satisfiability of Φ_1 can be checked in polynomial time, as Theorem 5.8 shows. Φ_2 consists entirely of universal formulae with no function symbols, and so if Φ_1^* is the result of saturating Φ_1 according to the proof method of Theorem 5.8, then the satisfiability of $\Phi_1^* \cup \Phi_2$, which is equivalent to the satisfiability of $\Phi_1 \cup \Phi_2$, can certainly be checked in NP.

To show NP-hardness, we reduce propositional satisfiability to satisfiability in Cop+TV+VCoord. Recall that it is sufficient to be able to decide the satisfiability of a set Φ of clauses of the forms $P \vee Q$, $\neg P \vee \neg Q$ and $\neg P \vee \neg Q \vee R$. Let c, d be

common nouns, and for every proposition symbol P occurring in such a set Φ, let p be a transitive verb. By introducing fresh common nouns p', q' where necessary, we translate elements of Φ into Cop+TV+VCoord-sentences as follows.

$P \vee Q$	c ps or qs d	$p(c,d) \vee q(c,d)$
$\neg P \vee \neg Q$	c does not p and q d	$\neg p(c,d) \vee \neg q(c,d)$
$\neg P \vee \neg Q \vee R$	c ps or p's d	$p(c,d) \vee p'(c,d)$
	c qs or q's d	$q(c,d) \vee q'(c,d)$
	c does not p and p' d	$\neg p(c,d) \vee \neg p'(c,d)$
	c does not q and q' d	$\neg q(c,d) \vee \neg q'(c,d)$
	c p's or q's or rs d	$p'(c,d) \vee q'(c,d) \vee r(c,d)$

These translations visibly preserve satisfiability, and so the problem of deciding satisfiability in Cop+TV+VCoord is NP-hard, and hence NP-complete. □

Theorem 6.18. *The satisfiability problem for Cop+TV+DTV+VCoord is NP-complete.*

Proof. Membership in NP follows by a nearly identical argument to that given above. NP-hardness is immediate, since every formula in Cop+TV+VCoord is also in Cop+TV+DTV+VCoord. □

Theorem 6.19. *Let F be one of the fragments Cop+Rel+TV, Cop+Rel+DTV or Cop+Rel+TV+DTV. Then F+VCoord has a NEXPTIME-complete satisfiability problem.*

Proof. To see membership in NEXPTIME, note that a set Φ of formulae in each fragment F+VCoord can be equisatisfiably rewritten, by introducing new predicates as necessary, as a set Φ_1 of F-formulae, and a set Φ_2 of purely universally-quantified, non-functional formulae of the forms

$$\forall x \forall y (t_1(x,y) \rightarrow t_2(x,y))$$
$$\forall x \forall y (t_1(x,y) \rightarrow t_2(x,y) \vee t_3(x,y))$$
$$\forall x \forall y \forall z (d_1(x,y,z) \rightarrow d_2(x,y,z))$$
$$\forall x \forall y \forall z (d_1(x,y,q) \rightarrow d_2(x,y,z) \vee d_3(x,y,z))$$

where t_1, t_2 and t_3 are binary relations and d_1, d_2 and d_3 are ternary relations.

In [58] and [60], it is shown that the satisfiability of Φ_1 can be checked in at worst NEXPTIME by saturation under resolution to give a set of clauses Φ_1^*. By the same argument used in the proof of Theorem 6.17, checking the satisfiability

of $\Phi_1^* \cup \Phi_2$ therefore cannot take more than NEXPTIME. Since $\Phi_1 \cup \Phi_2$ and $\Phi_1^* \cup \Phi_2$ are equisatisfiable, the result follows.

We need only show NEXPTIME-hardness for Cop+Rel+TV+VCoord. Theorems 5.30 and 5.35 ensure the NEXPTIME-hardness of Cop+Rel+DTV+VCoord and Cop+Rel+TV+DTV+VCoord.

By a slight adaptation of [60, Lemma 4.6], we can see that NEXPTIME-hardness follows if the satisfiability of clauses of the following forms can be reduced to satisfiability in Cop+Rel+TV+VCoord

$$\neg p(x) \vee \neg q(x) \vee \neg p(y) \vee \neg r(x) \vee \neg t_1(x, y)$$
$$\neg p(x) \vee q(x) \vee r(x)$$
$$\neg p(x) \vee t_1(x, f(x))$$
$$\neg p(x) \vee \neg p(y) \vee \neg t_1(x, y) \vee \neg t_2(x, y) \vee \neg t_3(x, y)$$
$$\neg p(x) \neg p(y) \vee t_1(x, y) \vee t_2(x, y)$$

where p, q, r are unary predicates, t_1, t_2, t_3 are binary relations and f is a unary (Skolem) function symbol.

Cop+Rel+TV+VCoord-formulae of the following forms suffice to carry out such a reduction:

$$\forall x(p(x) \wedge q(x) \rightarrow \forall y(p(y) \wedge r(y) \rightarrow \neg t_1(x, y)))$$
$$\forall x(p(x) \wedge \neg q(x) \vee r(x))$$
$$\forall x(p(x) \rightarrow \exists y(p(y) \wedge t_1(x, y)))$$
$$\forall x(p(x) \rightarrow \forall y(p(y) \wedge t_1(x, y) \wedge t_2(x, y) \rightarrow \neg t_3(x, y)))$$
$$\forall x(p(x) \rightarrow \forall y(p(y) \wedge t_1(x, y) \rightarrow t_2(x, y))$$

All of the above are Cop+Rel+TV-formulae, apart from the final two. But these are straightforwardly expressed in Cop+Rel+TV+VCoord with the equivalent sentences

No p t_1s and t_2s and t_3s any p
Every p t_1s or t_2s every p

Satisfiability in Cop+Rel+TV+VCoord is thus NEXPTIME-hard. □

6.7 Conclusion

We have presented an almost complete survey of the effects on semantic complexity of allowing various levels of coordination to the fragments introduced in Chapter 5. Table 6.7 summarises the results.

The obvious extension of the work in this chapter would be to continue as in Chapter 5 and provide semantic characterisations of each of these extended fragments. Although we have not yet looked into it, it might be expected that such results are easier to obtain than those in Chapter 5. After all, many of the complications in our earlier proofs arose from the very limited distribution of conjunction and disjunction in the fragments considered, compared to standard artificial logical languages. For the moment, however, we must leave such questions open.

Coordinated category	Fragment			
	Cop	Cop+TV	Cop+TV+DTV	Cop+Rel
IP	NP	NP	NP	NP
NP	NP	Open	Open	NP
N′	NP	Open	Open	NP
CP	N/A	N/A	N/A	NP
I′	NP	EXPTIME	NEXPTIME	NP
V	N/A	NP	NP	N/A

Coordinated category	Fragment		
	Cop+Rel+TV	Cop+Rel+DTV	Cop+Rel+TV+DTV
IP	EXPTIME	NEXPTIME	NEXPTIME
NP	EXPTIME	NEXPTIME	NEXPTIME
N′	EXPTIME	NEXPTIME	NEXPTIME
CP	EXPTIME	NEXPTIME	NEXPTIME
I′	EXPTIME	NEXPTIME	NEXPTIME
V	NEXPTIME	NEXPTIME	NEXPTIME

Table 6.1: Semantic complexity results for fragments with coordination

Chapter 7

Conclusions

Take care of the sense and sounds will take care of themselves.

— Lewis Carroll, *Alice's Adventures in Wonderland*

In this thesis, we have defined a range of simple fragments of English featuring basic constructions such as proper and common nouns, relative clauses, (di)transitive verbs and coordination, and assigned truth-conditional semantics to each. By treating each fragment as generating a corresponding fragment of first-order logic, we have been able to characterise the complexity of reasoning in each fragment, as well as the logical expressive power of those fragments without subsentential coordination. We have thereby provided criteria by which it is possible to determine when arbitrary expressions of first-order logic can be translated into certain fragments of English.

We have demonstrated techniques for carrying out precise analysis of the semantic contributions of syntactic constructions in natural language, using standard formalisms, and without any need for a custom "English-like" logical syntax. Such an approach allows the easy application of standard results and techniques of mathematical logic, with no wasted effort. The use of such standard techniques also has the advantage that this approach can generalise easily to any natural language.

The results presented here provide a detailed "map" of the interface between syntax and semantics, at least for some parts of English, and highlight some of the semantic limitations of some simple fragments of English. One of the most striking facts to have emerged is the extent to which logicians rely on arbitrary

conjunction and disjunction – a facility which, in the guise of relative clauses, and so on, is considerably more restricted in natural language. Such a contrast would be unlikely to be apparent if semantics were handled using a formalism with syntax closer to that of English.

As we noted in Chapter 2, the prevailing view seems to have been that translation of natural languages into "logicians' logics", such as first-order logic, is not really appropriate for studying questions related to natural language reasoning. Those who hold such a view seem to justify it by appeal to the claim that humans do not reason using, say, first-order syntax. However, the claim that humans reason using the *syntax* of their native language seems equally open to question. Regardless of the actual methods of human reasoning, the study of which we leave to psychology, given an accepted set of truth conditions for sentences, the potential models of the world in which they are true or false are fixed. As a consequence, so are relations such as entailment between sentences. We claim therefore that any semantic representation language which allows clear descriptions of such possible models is likely to be a useful tool for studying natural language semantics. We interpret the results and proofs of this thesis as justification for this claim.

The question of generality is also significant: a logical system whose syntax resembles that of English might well be easy enough to adapt for closely related language such as German, but could be hopeless for languages with drastically different syntactic structures, such as Latin or Czech, just to pick some examples even within the Indo-European family.

Several possible objections to the techniques we have used suggest themselves. Firstly, we have claimed that our particular choice of syntactic framework has no effect on the results, and that many of the syntactic shortcuts we have taken are similarly harmless. But could it not be that the same fragments, given in a different grammar formalism or using much more sophisticated syntactic restrictions, might have a different semantic complexity, for example? The answer seems to be no. Certainly it is straightforward to verify that each of our grammars do generate at least all grammatical sentences in the desired fragments. It is possible – indeed, probable – that they overgenerate, and produce sentences which ideally we would like to rule out. However, the upper bounds we have shown apply to the full, potentially overgenerated fragments produced by our grammars, and so would continue to hold even if the overgeneration were to be removed. The lower

bounds do not depend on all sentences generated by each grammar, but only on the particular sentences required for each hardness proof. A simple check of each such proof reveals that the sentences used in each case are all unquestionably grammatical, and do indeed express the truth-conditions assigned to them. Thus any grammar, in any formalism, purporting to generate the fragments concerned must generate those sentences, and assign those semantics. The lower bounds we have given for semantic complexity are also therefore immune to syntactic refinement.

The expressive power results are perhaps more fragile – it is entirely plausible that tighter definitions of fragments could require more precise notions of simulation to characterise their expressive power. We believe, however, that the essential details will turn out to be stable under refinement, and that our results capture the general semantic nature of the fragments we have studied.

Another objection which could be raised is that the language of our fragments is not really "everyday" English. People rarely produce sentences such as Every philosopher who is not a stoic is a cynic. While certainly true, it is nonetheless also true that the kinds of sentence we have used are extremely typical "defining" sentences, and a great deal of general world knowledge about, say, the denotations of various common nouns, and so on, can be expressed using them. Thus, although not necessarily examples of daily conversational English., the fragments we have defined *are* natural and useful candidates for study.

The semantic complexity results presented in this thesis contribute to the programme of work proposed in [58] and form an extension of that work. This programme is clearly far from complete, and no doubt much could be learned by studying further fragments of English, or naturally defined fragments of other natural languages.

The results on expressive power can likewise be extended to larger fragments, or to other languages. We have given here no characterisation of the expressive power of any of the fragments in Chapter 6, for example. In both [58] and [60], some of the fragments we have considered are extended with anaphora, admitting sentences such as Every cynic who admires some stoic respects him, subject to various restrictions on the denotations of anaphora. It would be interesting to provide expressive power results for these fragments too. A suggestion from an anonymous reviewer of [83] was to look into ways that the characterisation of expressive power might be automated, so that extension to larger fragments becomes easier. It is

unclear, at least at the moment, however, how such a task might be undertaken for fragments defined using recursive phrase structure rules.

As well as extending to larger or different fragments of English or other languages, further model-theoretic conclusions can be drawn from the expressive power results in this thesis. For example, it ought to be relatively trivial to give for a fragment F, a characterisation of the class of structures definable by a set of F-sentences, using techniques such as those used in [7, Chapter 3] where such results are presented for modal logic.

Finally, it seems likely that some of the results of Generalised Quantifier Theory, such as those discussed in Chapter 2, might offer some general insight into how to give semantic characterisations of fragments containing various quantifiers, and so could be used to develop further expressive power results of the kind given here.

Bibliography

[1] Hajnal Andréka, Johan van Benthem, and Istvan Németi. Back and forth between modal and classical logic. *Journal of the IGPL*, 3(5):685–720, 1995.

[2] Hajnal Andréka, Johan van Benthem, and Istvan Németi. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27:217–274, 1998.

[3] Emmon Bach, Eloise Jelinek, Angelika Kratzer, and Barbara H. Partee, editors. *Quantification in natural languages*, volume 54 of *Studies in Linguistics and Philosophy*. Kluwer Academic Publishers, Dordrecht, 1995.

[4] Jon Barwise and Robin Cooper. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4:159–219, 1981.

[5] Patrick Blackburn, Johan Bos, Michael Kohlhase, and Hans de Nivelle. Automated theorem proving for natural language understanding, 1998.

[6] Patrick Blackburn, Johan Bos, Michael Kohlhase, and Hans de Nivelle. Inference and computational semantics. In Harry Bunt and Elias Thijsse, editors, *Third International Workshop on Computational Semantics (IWCS–3)*, pages 5–21. 1999.

[7] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal logic*. Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, UK, 2001.

[8] E. Börger, E. Grädel, and Y. Gurevich. *The classical decision problem*. Perspectives in Mathematical Logic. Springer-Verlag, 1997.

[9] Robert D. Borsley. *Modern Phrase Structure Grammar*. Number 11 in Blackwell textbooks in linguistics. Blackwell Publishers, 1996.

[10] B. Carpenter. *Type Logical Semantics*. MIT Press, 1998.

[11] Maarten de Rijke and Holger Sturm. Global definability in basic modal logic. In H. Wansing, editor, *Essays on non-classical logic*, pages 111–135. World Scientific Publishers, 2001.

[12] Maarten de Rijke and Heinrich Wansing. Proofs and expressiveness in alethic modal logic. In D. Jacquette, editor, *A companion to philosophical logic*, pages 422–441. Blackwell, 2002.

[13] Devdatt P. Dubhashi. *Complexity of logical theories*. Number LS-95-5 in BRICS. University of Århus, Århus, Denmark, 1995.

[14] Heinz-Deiter Ebbinghaus and Jörg Flum. *Finite model theory*. Perspectives in Mathematical Logic. Springer-Verlag, 2nd edition, 1999.

[15] Frederic B. Fitch. Natural deduction rules for English. *Philosophical Studies*, 24:89–104, 1973.

[16] Robert W. Floyd and Richard Beigel. *The language of machines: an introduction to computability and formal languages*. W. H. Freeman, 1993.

[17] Yaroslav Fyodorov, Yoad Winter, and Nissim Francez. Order-based inference in natural logic. *Logic Journal of the IGPL*, 11(4):385–416, 2004.

[18] D. M. Gabbay and F. Guenthner, editors. *Handbook of philosophical logic*, volume 1. Reidel, 1983.

[19] D. M. Gabbay and F. Guenthner, editors. *Handbook of philosophical logic*, volume 2. Reidel, 1983.

[20] D. M. Gabbay, C. J. Hogger, and J. A. Robinson. *Handbook of logic in artificial intelligence and logic programming: epistemic and temporal reasoning*, volume 4. Oxford University Press, 1995.

[21] M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. W. H. Freeman, 1979.

[22] Robert Givan and David A. McAllester. Polynomial-time computation via local inference relations. *ACM Transactions on Computational Logic*, 3(4):521–541, 2002.

[23] E. Grädel. Decidable fragments of first-order and fixed-point logic: from prefix vocabulary classes to guarded logics. Invited talk at Kalmar Workshop on Logic and Complexity, Szeged, (commemorating the work of Laslo Kalmar), October 2003.

[24] Liliane Haegeman. *Introduction to government and binding theory*. Number 1 in Blackwell Textbooks in Linguistics. Blackwell Publishers, 2nd edition, 1994.

[25] I. Heim and A. Kratzer. *Semantics in generative grammar*. Number 13 in Blackwell textbooks in linguistics. Blackwell Publishers, 1998.

[26] J. Roger Hindley and Jonathan P. Seldin. *Introduction to combinators and λ-calculus*. Number 1 in London Mathematical Society Student Texts. Cambridge University Press, 1986.

[27] Colin Hirsch. *Guarded logics: algorithms and bisimulation*. PhD thesis, Rheinisch-Westfälische Technische Hochschule, 2002.

[28] Wilfrid Hodges. Logic and games. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Fall 2001.

[29] Wilfrid Hodges. First-order model theory. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Winter 2001.

[30] Wilfrid Hodges. Model theory. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Winter 2001.

[31] Ian Hodkinson. Finite variable logics. *Bulletin of the European Association of Theoretical Computer Science*, 51:111–140, 1993.

[32] Marco Hollenberg. Safety for bisimulation in monadic second-order logic. Logic Group Preprint Series 170, Dept. of Philosophy, Utrecht University, November 1996.

[33] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to automata theory, languages and computation*. Addison-Wesley, 1979.

[34] Jieh Hsiang and Michaël Rusinowitch. Proving refutational completeness of theorem-proving strategies: the transfinite semantic tree method. *Journal of the ACM (JACM)*, 38(3):558–586, 1991.

[35] Neil Immerman. Upper and lower bounds for first-order expressibility. *Journal of Computer and System Sciences*, 25:76–98, 1982.

[36] Neil Immerman and Dexter Kozen. Definability with bounded number of bound variables. *Information and Computation*, 83(2):121–139, 1989.

[37] A. Norman Jeffares. *W. B. Yeats: a new biography*. Continuum, 2001.

[38] H. Kamp and U. Reyle. *From discourse to logic: an introduction to model-theoretic semantics of natural language, formal logic and discourse representation theory*. Kluwer, Dordrecht, 1993.

[39] Edward Keenan and Jonathan Stavi. A semantic characterisation of natural language determiners. *Linguistics and Philosophy*, 9:253–326, 1986.

[40] Edward L. Keenan, editor. *Formal semantics of natural language*. Cambridge University Press, 1975.

[41] Edward L. Keenan. Some properties of natural language quantifiers: generalized quantifier theory. *Linguistics and Philosophy*, 25:627–654, 2002.

[42] Edward L. Keenan and Lawrence S. Moss. Generalized quantifiers and the expressive power of natural language. In Johan van Benthem and Alice ter Meulen, editors, *Generalized quantifiers in natural languages*, volume 4 of *GRASS*, chapter 4, pages 74–124. 1985.

[43] Phokion G. Kolaitis. On the expressive power of logics on finite models. In *Finite model theory and its applications*, EATCS: Texts in Theoretical Computer Science. Springer, 2003. to appear.

[44] Phokion G. Kolaitis and Moshe Y. Vardi. On the expressive power of variable-confined logics. In *LICS '96: Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, page 348. IEEE Computer Society, 1996.

[45] Alexander Leitsch. *The resolution calculus*. Texts in Theoretical Computer Science. Springer-Verlag, 1997.

[46] David A. McAllester and Robert Givan. Natural language syntax and first-order inference. *Artificial Intelligence*, 56(1):1–20, 1992.

[47] James D. McCawley. *The syntactic phenomena of English*. University of Chicago Press, Chicago, 2nd edition, 1998.

[48] Stefan Müller. HPSG bibliography. `http://www.dfki.de/lt/HPSG/hpsg_bib.html`.

[49] Seungho Nam. n-ary quantifiers and the expressive power of DP compositions. *Research on Language and Computation*, 3, 2005.

[50] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[51] Hester Lynch Thrale Piozzi. *Anecdotes of the late Samuel Johnson*. Cadell of London, 1786.

[52] Bruno Poizat. *A course in model theory: an introduction to contemporary mathematical logic*. Springer-Verlag, 2000.

[53] Carl J. Pollard and Ivan A. Sag. HPSG: A new theoretical synopsis. In Byung-Soo Park, editor, *Linguistic Studies on Natural Language*, volume 1 of *Kyunghee Language Institute Monograph*. Hanshin, Seoul, Korea, 1992.

[54] Carl J. Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago, 1994.

[55] Ian Pratt and David Brée. The expressive power of the English temporal preposition system. Technical Report UMCS-93-1-7, Dept. of Computer Science, University of Manchester, 1993.

[56] Ian Pratt-Hartmann. On the semantic complexity of some fragments of English. Technical Report UMCS-00-5-1, University of Manchester, 2000.

[57] Ian Pratt-Hartmann. A two-variable fragment of English. *Journal of Logic, Language and Information*, 12:13–45, 2003.

[58] Ian Pratt-Hartmann. Fragments of language. *Journal of Logic, Language and Information*, 13:207–223, 2004.

[59] Ian Pratt-Hartmann. Temporal prepositions and their logic. *Artificial Intelligence*, 166(1–2):1–36, 2005.

[60] Ian Pratt-Hartmann and Allan Third. More fragments of language. *Notre-Dame Journal of Formal Logic*, 2004. forthcoming.

[61] William C. Purdy. Expressiveness of fluted logic.

[62] William C. Purdy. A logic for natural language. *Notre Dame J. Formal Logic*, 32(3):409–425, 1991.

[63] William C. Purdy. Surface reasoning. *Notre Dame J. Formal Logic*, 33(1):13–36, 1992.

[64] William C. Purdy. A variable-free logic for mass terms. *Notre Dame J. Formal Logic*, 33(3):348–358, 1992.

[65] William C. Purdy. Fluted formulas and the limits of decidability. *J. Symbolic Logic*, 61(2):608–620, 1996.

[66] William C. Purdy. Surrogate variables in natural language. In Michael Boettner and Wolf Thuemmel, editors, *Variable-free semantics*, volume 3. 2000.

[67] William C. Purdy. Complexity and nicety of fluted logic. *Studia Logica*, 71(2):177–198, 2002.

[68] Susan Ratcliffe, editor. *The little Oxford dictionary of quotations*. OUP, 1994.

[69] John Alan Robinson and Andrei Voronkov, editors. *Handbook of Automated Reasoning*. Elsevier and MIT Press, 2001.

[70] John Robert Ross. *Infinite syntax!* Erlbaum, 1986.

[71] I. Susan Russinoff. The syllogism's final solution. *Bulletin of Symbolic Logic*, 5(4), 1999.

[72] Ivan A. Sag and Carl J. Pollard. Head-driven phrase structure grammar: An informal synopsis. CSLI Report 87-79, Stanford University, Stanford University, 1987.

[73] Fred Sommers. *The logic of natural language*. Clarendon Press, 1982.

[74] Fred Sommers. Predication in the logic of terms. *Notre-Dame Journal of Formal Logic*, 31(1):106–126, 1990.

[75] Mark Steedman. A very short introduction to CCG. `http://www.iccs.informatics.ed.ac.uk/~steedman/papers.html`.

[76] Mark Steedman. Categorial grammar. In Rob Wilson and Frank Keil, editors, *The MIT Encyclopaedia of Cognitive Sciences*. MIT Press, 1998.

[77] Mark Steedman. *The syntactic process*. MIT Press/Bradford Books, 2000.

[78] Mark Steedman and Jason Baldridge. Combinatory categorial grammar.

[79] Patrick Suppes. Logical inference in English: a preliminary analysis. *Studia Logica*, 38(4):375–391, 1979.

[80] Anna Szabolcsi, editor. *Ways of scope taking*. Kluwer Academic Publishers, Dordrecht, 1997.

[81] Tanel Tammet. Using resolution for deciding solvable classes and building finite models. In *Lecture Notes in Computer Science*, volume 502, pages 33–64. Springer, 1991.

[82] Tanel Tammet. *Resolution methods for decision problems and finite-model building*. PhD thesis, Chalmers University of Technology, Göteborg, Sweden, 1992.

[83] Allan Third. The expressive power of restricted fragments of English. In Philippe Blache, Edward Stabler, Joan Busquets, and Richard Moot, editors, *Logical Aspects of Computational Linguistics 2005*, volume 3492 of *Lecture Notes in Artificial Intelligence*, pages 317–329. Springer-Verlag, 2005.

[84] R. Thomason, editor. *Formal philosophy: selected papers of Richard Montague*. Yale University Press, New Haven, 1974.

[85] Jouko Väänänen and Dag Westerståhl. On the expressive power of monotone natural language quantifiers over finite sets. *Journal of Philosophical Logic*, 31:327–358, 2002.

[86] Johan van Benthem. *Modal correspondence theory*. PhD thesis, Mathematisch Instituut and Instituut voor Grondslagenonderzoek, University of Amsterdam, 1976.

[87] Johan van Benthem. Correspondence theory. In Gabbay and Guenthner [19], pages 167–247.

[88] Johan van Benthem. Questions about quantifiers. *Journal of Symbolic Logic*, 49(2):443–466, 1984.

[89] Johan van Benthem. Meaning: interpretation and inference. *Synthese*, 73:451–470, 1987.

[90] Johan van Benthem. *Exploring logical dynamics*. CSLI Publications, Stanford, 1996.

[91] Johan van Benthem. Modality, bisimulation and interpolation in infinitary logic. *Annals of Pure and Applied Logic*, 96:29–41, 1999.

[92] Geoffrey Willans and Ronald Searle. *Molesworth*. Penguin Classics, 1999.

[93] Mary McGee Wood. A categorial syntax for coordinate constructions. Technical Report UMCS-89-2-1, University of Manchester Department of Computer Science, 1989.