

CHAPTER 1

INTRODUCTION

LOGIC is a powerful set of tools which have found applications in many areas of artificial intelligence. The Lambek calculus (Lambek 1958) and its extensions and generalizations (Morrill 1994, van Benthem 1995, Moortgat 1997) have been the principal systems used for giving a logical account of natural language.

The advantages of a logical view of grammar are manifold.

- we can prove *soundness* of our system, that is we can show all operations in our grammar are meaningful and well-defined with respect to a model.
- we can prove *completeness* of our system, that is we can show that if something is valid in our model, it is therefore derivable in our logic as well.
- we can prove *consistency* of our system. This means that it is fundamentally impossible in our logic that a statement and its negation are both true.

Of course, we can also prove the correctness of algorithms given a precise mathematical specification of the preconditions and the postconditions of the algorithm, as advocated by Dijkstra (1979). However, in this case, changes in the algorithm will force us to reprove correctness, whereas for a logic we prove soundness and completeness of the logic itself and if we should choose to apply this logic to a different domain no new proofs will be necessary.

Figure 1.1 gives an overview of the logical approach to grammar. A parser for a logical grammar consists of a lexicon, which assigns sets of formulas to words and a theorem prover which proves that statements in the logic are either provable or unprovable.

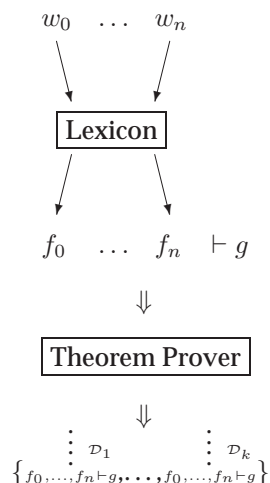


Figure 1.1: A schematic overview of the logical approach to grammar

The slogan here is *parsing as deduction*. This term was introduced by Pereira, Warren and others (Pereira & Warren 1983), (Pereira & Warren 1980), who used the definite clause grammar facilities of the general purpose logic programming language Prolog as a logical way of describing grammars. A good overview of using Prolog for natural language analysis can be found in (Pereira & Shieber 1987) and (Shieber, Schabes & Pereira 1995).

Our goal here is more ambitious, instead of using a general purpose logic to give a description of our grammar and run this description as a program, we want to use a logic which is inherently suitable for linguistic descriptions. We are interested in what Moortgat calls the ‘constants of grammatical reasoning’ (Moortgat 1999); the logical universals when we reason about grammars.

A Lambek calculus theorem prover should return a *set* of proofs: the set of different proofs for the given logical statement. A potential problem here is that we are only interested in proofs which are different for interesting reasons, as opposed to proofs which differ only because of an overly bureaucratic proof system. The introduction of a ‘redundancy free’ formulation of proofs, proof nets, will solve this problem for us. Different proof nets are semantically different linguistic objects. In Section 3.5, we will see that there is a very general way of assigning lambda term semantics to Lambek calculus proofs, which is an attractive property of Lambek calculi and it is one we would be a shame to lose this property because of the syntactic peculiarities of our proof system. Finding and analysing proof net calculi suitable for linguistic analysis will be the central theme of this book.

Though we advocate a lexicalist account of grammar we do not want to shift too much of the explanatory burden to the lexicon. As shown by Carpenter (1991), adding recursive lexical rules to even a very simple grammar results in an undecidable system. Therefore, we will assume that the lexicon

assigns a finite number of formulas to every word in the lexicon.

We also want to avoid assigning formulas to the empty string, because this will increase the expressiveness of the system in much the same way as adding lexical rules does; every logical formula we pass to the theorem prover is keyed to a word in the input sentence.

1.1 Overview

This book is structured as follows.

Part I is an introduction to the use of logic for grammatical analysis.

Chapter 2 will introduce linear logic as a first approximation and suggest linguistic applications for the different groups of logical connectives. A clear limitation of linear logic is the global availability of the structural rule of commutativity, which incorrectly predicts any permutation of a grammatical sentence is also grammatical.

In Chapter 3 we will introduce the Lambek calculus, a non-commutative precursor of linear logic, and show how the addition of multiple families of connectives and of unary modal connectives increases the linguistic sophistication of the theory.

Part II is the main part of this book and focuses on how to use proof nets for linguistic analysis.

Chapter 4 introduces proof nets as a redundancy free representation for proofs in multiplicative linear logic and for proofs in the associative Lambek calculus.

Chapter 5 consists of joint work with Mario Piazza, which has appeared before in (Moot & Piazza 2001). We will study proof nets for the first order fragment of multiplicative linear logic and give an embedding translation for both the associative and the non-associative Lambek calculus into the first order fragment. We will also see how to use the first order fragment for a treatment of linguistic phenomena which have no satisfactory treatment in the Lambek calculus.

In Chapter 6 we will look at labeled deduction as a way of adding structural constraints to our logical calculi and to the proof nets for multiplicative linear logic.

Chapter 7 consists of joint work with Quintijn Puite, and large parts of it have appeared before in (Puite & Moot 1999, Moot & Puite 1999, Moot & Puite 2001), contains the main result of this book: by reformulating proof nets in a more symmetric way, we give a proof net system for the multimodal Lambek calculus and prove it is sound and complete with respect to the sequent formulation of the same calculus.

In Part III we will discuss some computational implications of the proof net calculus of Chapter 7 and relate it to other formalisms.

Chapter 8 will be devoted to automated deduction. We will present an algorithm for proof search using proof nets and some heuristics for improving the performance of this algorithm.

In Chapter 9 we will establish a PSPACE complexity result for the multimodal Lambek calculus with a restriction of the form of the structural rules and show that without this restriction the logic is undecidable.

Chapter 10 will give an embedding of Lexicalized Tree Adjoining Grammars into multimodal proof nets.

Finally, Chapter 11 contains my concluding remarks and a discussion of possible future research.

An Appendix gives an introduction to the Grail automated theorem prover for the multimodal Lambek calculus, which was developed as part of this research project.