

THROUGHOUT this book, we have seen how proof nets in their various forms can be used as a logical tool for the description of linguistic analyses. The main advantage of proof nets over other formulations of the multimodal Lambek calculus, such as natural deduction or sequent calculi, is that proof nets are inherently redundancy free, that is, different proof nets correspond to different linguistic objects, they are not merely different representations of the same linguistic object. In this sense, proof nets capture the ‘essence’ of proofs.

This property is not merely conceptually attractive, but also offers computational advantages in that we do not need to formulate procedural restrictions on the order of rule applications in the system to avoid generating duplicate solutions, or worse, *check* every time we find a solution to see if it is a new solution or a syntactic variant of an old solution.

We have given proof net calculi for **MLL**, \mathbf{L}_e , **MLL1** and finally for $\mathbf{NL} \diamond_{\mathcal{R}}$ and looked at correctness criteria for these calculi in the form of graph contractions, switchings and labeling. We have also given several possible linguistic applications of these calculi.

The logic $\mathbf{NL} \diamond_{\mathcal{R}}$, because it contains a variable structural rule component \mathcal{R} , was given a modular correctness criterion where the structural rules of \mathcal{R} correspond to graph rewrite rules and where the logical rules correspond to special cases of the graph contractions for **MLL**. We have given an algorithm for checking this contraction criterion and analyzed its complexity.

Finally, we have presented the Grail grammar development tool, which is based on this algorithm and which includes many of the heuristics we discussed in Chapter 8. This system has been used as courseware for introductory to advanced level courses in linguistics and artificial intelligence.

11.1 Further Research

Some questions were left open during throughout the chapters of this book.

First of all, though PSPACE is a good upper bound on the complexity for multimodal Lambek calculi, a similar characterization of NP complete and polynomial fragments would be useful. I have suggested the use of special cases of the contraction criterion for NP decidability and of extensions of the LTAG translation for polynomial decidability.

Instead of restricting the logic, another approach would be to *extend* the logic with operators like second order quantifiers or a contraction modality. We have suggested some uses for these connectives in Chapter 2, though it is still unclear if we can give an account of the linguistic phenomena discussed there without using the exponentials or second order quantifiers. Adding these connectives to the logic will be a delicate undertaking, since it can have a devastating effect on the complexity of the resulting logic. Proposals for decidable extensions of the Lambek calculus with a contraction modality or second order quantifiers are given by Jäger (2001) and Emms (1993*b*) respectively. Also, it would be interesting to see what proof nets for these logics would look like.

Finally, not all of the algorithmic improvements discussed in Chapter 8 have been implemented in the Grail automated theorem prover we will discuss in Appendix A. Measuring the effect of the different heuristics on performance and discovering other, more powerful heuristics will be useful for practical applications of the system.