THE UNIVERSITY *of* EDINBURGH

Automated Proof Search in Non-Classical Logics:
Efficient Matrix Proof Methods for Modal and Intuitionistic Logics

Lincoln A. Wallen

Ph.D.

University of Edinburgh

1987

Acknowledgements

I would like to thank the following people:

- my supervisor Alan Bundy for inspiration and advice during the research and throughout the preparation of this thesis;

- Paul Brna, Roberto Desimone, Jane Hesketh, Helen Pain and Alan Smaill for advice, proof-reading and support;

- Paul Brna and Jane Hesketh for their invaluable help during the tricky process of submission;

- past and present members of the Mathematical Reasoning Group for creating a stimulating and supportive environment to work in;

- and especially Sonya MacAngus for love and support throughout the years.

The research reported in this thesis was supported by an SERC studentship.

Published Papers

Some of the material contained in this thesis has been published in the following papers:

- Generating connection calculi from tableau- and sequent-based proof systems. In A.G. Cohn and J.R. Thomas, editors, *Artificial Intelligence and its Applications*, pages 35–50, John Wiley & Sons, 1986. Proceedings of AISB85, Warwick, England, April 1985.

- Formulating proof systems for automated deduction. In *Proceedings of the IEE Colloquium on Theorem Provers in Theory and Practice*, March 1987.

- A computationally efficient proof system for S5 modal logic (with G.V. Wilson). In J. Hallam and C. Mellish, editors, *Advances in Artificial Intelligence*, pages 141–153, John Wiley & Sons, 1987. Proceedings of AISB87, Edinburgh, Scotland, April 1987.

- Matrix proof methods for modal logics. In J. McDermott, editor, *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 917–923, Morgan Kaufmann Inc., 1987.

Copies of these papers may be found inside the back cover of this thesis.

Abstract

In this thesis we develop efficient methods for automated proof search within an important class of mathematical logics. The logics considered are the varying, cumulative and constant domain versions of the first-order modal logics K, K4, D, D4, T, S4 and S5, and first-order intuitionistic logic. The use of these non-classical logics is commonplace within Computing Science and Artificial Intelligence in applications in which efficient machine assisted proof search is essential.

Traditional techniques for the design of efficient proof methods for classical logic prove to be of limited use in this context due to their dependence on properties of classical logic not shared by most of the logics under consideration. One major contribution of this thesis is to reformulate and abstract some of these classical techniques to facilitate their application to a wider class of mathematical logics.

We begin with Bibel's Connection Calculus: a matrix proof method for classical logic comparable in efficiency with most machine orientated proof methods for that logic. We reformulate this method to support its decomposition into a collection of individual techniques for improving the efficiency of proof search within a standard cut-free sequent calculus for classical logic. Each technique is presented as a means of alleviating a particular form of redundancy manifest within sequent-based proof search. One important result that arises from this anaylsis is an appreciation of the rôle of unification as a tool for removing certain proof-theoretic complexities of specific sequent rules; in the case of classical logic: the interaction of the quantifier rules.

All of the non-classical logics under consideration admit complete sequent calculi. We anaylse the search spaces induced by these sequent proof systems and apply the techniques identified previously to remove specific redundancies found therein. Significantly, our proof-theoretic analysis of the rôle of unification renders it useful even within the propositional fragments of modal and intuitionistic logic.

The result is a set of matrix proof methods for the modal and intuitionistic logics considered, that, we argue, are more efficient than the other proof methods suggested in the literature for this class of first-order logics. We are careful in our generalisation of the matrix method so that search strategies developed for use with Bibel's classical matrix method remain applicable in conjunction with our non-classical matrix methods.

In summary: we successfully formulate demonstrably efficient matrix proof methods for automated proof search within a comprehensive class of first-order modal logics and first-order intuitionistic logic. In so doing we isolate general techniques for the design of efficient proof systems which may be applicable to other classes of mathematical logic encountered in the future.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction.

1.1 The problems and their solution.

The research reported in this thesis is concerned with the automation of proof search within mathematical logics. It lies in the area of automated theorem proving (ATP). In this context, our main problem is to formulate efficient methods for automated proof search within an important class of non-classical logics comprising:

- the modal logics: K, K4, D, D4, T, S4 and S5, and
- intuitionistic logic.

These logics, and their derivatives, are in widespread use within Computing Science and Artificial Intelligence, mostly in applications that require efficient methods of proof search.

Whilst our particular problem is easily stated, there are suprisingly few methods immediately to hand for its satisfactory solution. ATP has traditionally concentrated on the automation of proof search within classical logic, hence the emergence of the term "non-classical" to denote logics that differ semantically from classical logic. Not only has attention centered on one particular logic, but

also on a collection of techniques built around refinements of Robinson's resolution inference system for that logic [Rob65]. Unfortunately these techniques are not directly applicable to non-classical logics because they require a particular normal-form which, in general, does not exist for such logics.

Part of our solution is to look outside of the resolution paradigm and develop more abstract techniques that are not dependent on the properties of one particular logic. We note that a suitable level of abstraction does exist since all of the logics we are interested in admit straightforward sequent proof systems (see *eg.*, [Fit83]). Sequent calculi were originally developed by Gentzen [G69] as proof-theoretic tools for the analysis of proof within classical and intuitionistic logic. We start with an efficient proof method, and a cut-free sequent calculus for classical logic. The efficient proof method is Bibel's Connection Calculus [Bib82a,Bib82c]: a highly efficient matrix characterisation of validity for classical logic. We analyse the proof search space induced by the sequent calculus and identify three classes of redundancy within it. The classification of the redundancies is not based on any features pertaining particularly to classical logic. We then (re)formulate the Connection Calculus as a collection of individual, theoretically motivated, techniques for removing these particular redundancies from the search space. The net result is the identification of fairly general techniques for the alleviation of certain sequent-based redundancies. We have the beginnings of a general theory of efficient proof system design.

We then repeat the analysis, starting with cut-free sequent calculi for the modal logics under consideration. We identify similar redundancies in the modal search space as appeared in the classical case. We proceed to remove them by adapting the appropriate techniques identified in the original analysis. The result is a set of matrix characterisations of validity for the propositional and first-order versions of the modal logics K, K4, D, D4, T, S4 and S5, including their varying, cumulative and constant domain variants (a total of 20 distinct modal logics in all).

Next we turn to intuitionistic logic and repeat the derivation. Once again we start with a cut-free sequent calculus and analyse the redundancies in the

induced search space. The formulation of a matrix characterisation of validity for intuitionistic logic follows in the same manner as for the modal logics.

The resulting characterisations support proof search in exactly the same way as Bibel's original characterisation of classical logic. Consequently, the search methods that he and Andrews have already developed carry over *without change* to the modal and intuitionistic systems [And81,Bib81,Bib82b]. In [Bib82b], Bibel compares these methods (for classical logic) favourably with the most efficient refinements of resolution. His results serve to demonstrate the degree of efficiency we have achieved for the automation of proof search within the non-classical logics under consideration.

One slightly unexpected contribution of our analysis of the problems of proof search in modal and intuitionistic logics is that we are able to give a comprehensive classification of the redundancies within the alternative proof systems proposed in the literature for automated proof search in these logics. It turns out that all of them contain redundancies not shared by our matrix characterisations (and some contain other, more serious problems as well).

To summarise: our contribution is two-fold,

- We have succeeded in formulating efficient matrix proof methods for an important class of non-classical logics, and have therefore solved our main problem.

- We have identified powerful techniques for improving the efficiency of sequent-based proof procedures in general. These techniques can be applied individually when and where the prerequisite conditions apply, in particular, to develop efficient methods of proof search in non-classical logics.

We believe that this research is an important contribution to the field of automated theorem proving.

In the rest of this introductory chapter we consider the elements of the above argument that we do not concern ourselves with in the main body of this thesis.

In §1.2, we present the motivation for our interest in the efficient automation of proof search within non-classical logics in general, and modal and intuitionistic logics in particular. In §1.3, we define for the purposes of this thesis what is meant by the "efficiency" of proof procedures for a given logic, and how that efficiency may be improved. Since we do not go into the detail of proof procedures based on the matrix characterisations of validity developed in the thesis, in §1.4 we motivate our interest in this type of characterisation as a basis for practical proof procedures. For those readers not familiar with matrix-based proof procedures we include a short description of a simple example. Finally, in §1.5 we outline the structure of this thesis.

1.2 Problem solving by computer.

Mathematical logic is an important tool for both the theory and practice of Computing Science and Artificial Intelligence. Analytical techniques from logic form the basic toolkit for the mathematical investigation of computational concepts ranging from the semantics of programming languages (*eg.*, [GMW79,Mar82]), through the properties of distributed systems (*eg.*, [Sti85b,HM84]), the integrity and structure of databases (*eg.*, [Gra84]), the representation of knowledge and belief [Kon86,HM85], to the semantics of natural language [DWP81]. In many of these applications, logic is not only used as a mathematical tool for analysis but also as a practical tool for the representation and processing of information (data). Our concern in this thesis is precisely this practical use of mathematical logic as a formalism for representing and solving problems using a computer.

Mathematical logic, as a branch of mathematics, is primarily concerned with the notion of *logical consequence*. A logic typically consists of a formal language for writing sentences to represent information, and a consequence relation, $\models$, between sentences and sets of sentences. $\Gamma \models A$ expresses the fact that the sentence A "follows from," or is a "logical consequence of," the set of sentences Γ. If A follows from the empty set of sentences, we say A is *valid* in the logic.

Consequence relations are usually defined abstractly via an interpretation of the formal language in some mathematical structure. This structure forms the *semantics* of the logic (see *eg.*, [CK73]).

There is a powerful method of solving problems using a combination of mathematical logic and a computer. Suppose we are given a problem to solve in some domain. Suppose also that we have an algorithm that can determine whether or not a given sentence of the logic is valid. If we can represent our problem as a sentence of the logic in such a way that the original problem is solved if and only if the sentence representing it is valid, then we can solve our problem by executing the validity checking algorithm with the sentence as input. This problem solving method is utilised extensively in the applications mentioned above. Deductive problems can obviously be solved in this way, but even problems such as program synthesis can be transformed into such an inferential paradigm (see *eg.*, [MW80,Mar82,Con86]). In fact, one could argue that by viewing the logical language as a programming language, and the validity checking algorithm as an interpreter, this problem solving method is simply a generalisation of programming itself [Kow79].

The problem solving method outlined above is based on two suppositions:

1. that we can faithfully represent our informal problem as a sentence of the logic, and

2. that we have a suitable algorithm for checking the validity of sentences of the logic.

Unfortunately, these two criteria are antagonistic.

The representation of a complex problem within a given domain as a sentence (or sentences, if our algorithm checks logical consequence) of a formal logic is not an easy task. The difficulty lies in ensuring that the sentence is a faithful representation of the problem and the domain. This is eased if the semantics of the logic captures some general properties of the domain. For example, suppose

our domain required reasoning about information flow through a complex system and concerned the information available to components of the system. If the formal language of the logic chosen to represent the problem contains operators whose semantics mirror the way in which information can flow throughout the system we will be able to interpret sentences in the language informally, enhancing our ability to judge whether the formalisation of the problem is faithful. The representation of our problems within such a language will therefore be conceptually easier than if the semantic structure of the logic bore no resemblance whatsoever to the structure of the domain. (This is analogous to the ease a programmer might have in implementing a given recursive function in a programming language with explicit recursion, compared to a language without it.)

To summarise: in order to facilitate the correct representation of domains and problems we need logics whose semantics captures general properties of those domains. Typically then, each domain will require a specific logic. Turner [Tur84] summarises a large number of domain-specific logics developed or adapted for such representational use in Computing Science and Artificial Intelligence. Classical logic is just one such logic. Non-classical logics are the norm rather than the exception for such applications. Of course classical logic, being particularly general, can be used in situations where its semantic basis is not appropriate by *encoding* the structure of the domain explicitly. Following the programming language analogy: we could code our solutions to complex problems in a uniform assembly language; classical logic being that assembly language.

Unfortunately, as soon as the semantics of a logic becomes non-trivial, checking sentences for validity in the logic becomes difficult. Practically though, we can still solve problems in the manner outlined above provided we are careful about the way we formulate the algorithms. This is the main topic addressed in this thesis in relation to modal and intuitionistic logics.

Modal logics, in particular, are used extensively in various branches of Artificial Intelligence and Computing Science as logics of knowledge and belief (*eg.*, [Moo80,HM85,Kon84]), logics of programs (*eg.*, [Har79,Pne77]), and for such

tasks as the specification of distributed and concurrent systems (*eg.*, [HM84, Sti85b]). In many — if not all — of these applications the need arises for proof systems which facilitate efficient automated proof search.

Modal logics are extensions of classical logic obtained by including the (unary) modal operators $\Box$ and $\Diamond$, the operators of necessity and possibility respectively. They arose from a desire to formalise notions such as possibility. Hughes and Cresswell's book [HC68] is a good introduction to the field of modal logics.

Whilst modal logics are extremely popular currently, the author's original technical goal was to formulate an efficient proof procedure for intuitionistic logic [Dum77]. This logic (and other formal systems with an intuitionistic basis) has been identified as perhaps the central logic for capturing basic computational constructs. The crucial semantic notion that they formalise is that of "construction" as can be seen from the intuitionistic interpretation of implication [Dum77]:

- p is a proof of $A \Rightarrow B$ iff it is an effective operation that takes any proof of A into a proof of B.

As motivation for the material presented in this thesis, intuitionistic logics are being proposed as logics for *program derivation*. The problem of constructing an algorithm that satisfies a given specification is mapped onto the problem of proving the specification within an intuitionistic logic (see *eg.*, [Mar82,Con86]). The logics are usually typed, and a proof of the sentence:

$$\forall x \in A.\exists y \in B.R(x,y)$$

within these intuitionistic logics defines a function, f, that takes an element a, of the type A, to an element $f(a)$ of the type B, such that $R(a, f(a))$ holds. Although the design of algorithms is a difficult task, some aspects of this theorem-proving process are amenable to automation. For that, we need efficient proof methods for automated proof search in such logics.

1.3 The efficiency of proof procedures.

In this section we define what we mean by the "efficiency" of a proof procedure, and outline the criteria by which we relate different proof procedures in the sequel.

There are currently no general methods for comparing the efficiency of two arbitrary proof procedures. The situation resembles the comparison of the efficiency of different programming languages. The procedures may run on different machines, or may require the setting of parameters, or the input of the problem in a certain form, and so on. Typically, benchmark problems are used to provide a primitive method of comparison.

Following Meltzer [Mel71], we can decompose a proof procedure into two components:

- an *inference system*, and
- a *search strategy*.

Inference systems are simply calculi, or refinements of calculi. For example, the sequent calculus is an inference system for classical logic; so is the resolution rule of inference. Given a sentence, an inference system induces a search space of legal inference steps in which may lie proofs (or more generally: demonstrations of the validity) of the sentence, if it is valid. We can define the completeness of an inference system in the usual way, in terms of whether or not every valid sentence has a proof in the search space induced by the inference system.

Search strategies are methods of traversing the space induced by an inference system. We can talk about the completeness of a search strategy in terms of whether or not it searches the entire search space.

In this thesis we are primarily concerned with inference systems. Our goal is to define inference systems that induce as small a search space as possible, and

yet still remain complete. When we talk of "redundancies in a search space" we mean that the space contains a class of interior nodes (derivations or some other form of intermediate state) which can be uniformly eliminated by altering the inference system without losing completeness. If a suitable modification is made, the search space induced by the new inference system on an arbitrary problem is wholly contained within the search space induced by the old inference system for that problem. We say the new system is more efficient than the old. We talk of a redundancy being "removed" from the old system to form the new one.

We could potentially run into problems in trying to relate the matrix characterisations to the other inference systems proposed in the literature for modal and intuitionistic logics. However,

- The sequent/tableau proof systems [Fit83,Fit69] are easy to relate to the matrix characterisations since we show explicitly how the latter are refinements of the former.

- The clausal resolution systems [Far82,Far83,Far86] do not treat the full modal language and so are flawed in that respect. Moreover, in [Bib82b] Bibel carefully compares refinements of clausal resolution with various proof procedures based on the matrix characterisations. He defines procedures that simulate the most efficient of the clausal resolution techniques.

- The non-clausal resolution systems developed for these logics [AM86a, AM86b] constitute the most comprehensive extension of resolution-based ideas to modal logics. However, the use of resolution is restricted and explicit deduction rules rules are used to manipulate the modal operators. These proof systems turn out to induce extremely redundant search spaces. We describe how these redundancies arise mainly from the failure to deal with the order dependence of the modal rules.

- The hybrid resolution/tableau systems [Kon84,Kon86] are not effective, in that arbitrary search is required to determine the correctness of a single

inference. The relationship of these systems with the matrix characterisations is easily established due to their basis in tableaux.

In fact, using our analysis of the redundancies in sequent-based proof search we are able to give a comprehensive classification of the redundancies within these alternative proof systems.

1.4 Matrix-based proof search.

In this section we motivate our interest in extending classical matrix methods to non-classical logics, as opposed to other styles of proof procedure.

1.4.1 Matrices, paths and connections.

The basic idea of the matrix methods, dating back to Prawitz' procedures [Pra60], is that a formula of classical logic can be represented as a two-dimensional matrix. For example, the formula:

$$(P \Rightarrow Q) \wedge (Q \Rightarrow R) \Rightarrow (P \Rightarrow R)$$

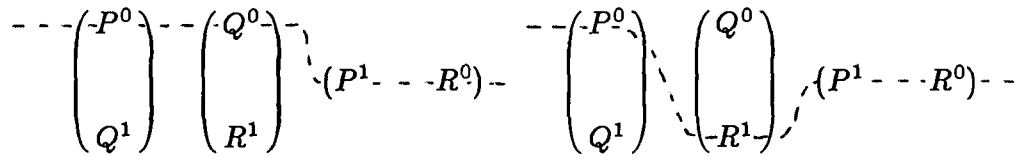
can be seen as the matrix:

$$\begin{pmatrix} P^0 \\ \Downarrow \\ Q^1 \end{pmatrix} \quad \wedge \quad \begin{pmatrix} Q^0 \\ \Downarrow \\ R^1 \end{pmatrix} \quad \Rightarrow \quad (P^1 \Rightarrow R^0)$$

Or, omitting the connectives:

$$\begin{pmatrix} P^0 \\ \\ Q^1 \end{pmatrix} \quad \begin{pmatrix} Q^0 \\ \\ R^1 \end{pmatrix} \quad (P^1 \qquad R^0)$$

This matrix consists of four columns. There is a natural notion of path through such a matrix. A path is a set of atomic elements, one from each column of the matrix. Two such paths are shown below.



One way of identifying the matrix representation of a formula is eliminate all connectives except conjunctions and disjunctions, and push the negation symbols down to the atomic level. This equivalent form of any classical formula is called its *negation normal-form.* The negation normal-form of our example formula is:

$$(P \wedge \neg Q) \vee (Q \wedge \neg R) \vee (\neg P \vee R)$$

If we now replace a negated atom $\neg P$ by P^1, and an unnegated atom P by P^0 we get:

$$(P^0 \wedge Q^1) \vee (Q^0 \wedge R^1) \vee (P^1 \vee R^0)$$

The superscript of an atom is called its *polarity.*

Finally, placing the components of a conjunction vertically, and the components of a disjunction side-by-side, and omitting the connectives, we obtain the matrix given above. Since disjunction is associative and commutative, we can reorganise the columns (disjuncts) of the matrix thus:

$$P^1 \quad \begin{pmatrix} P^0 \\ \\ Q^1 \end{pmatrix} \quad \begin{pmatrix} Q^0 \\ \\ R^1 \end{pmatrix} \quad R^0$$

Notice that this does not affect the contents of the paths through the matrix. It should be noted that the matrix is a human aid for describing the method. No normal-forming is needed to define the paths through it.

A *connection* (hence: "Connection Calculus") is a pair of atomic formula occurrences in some path with different polarities, *i.e.*, $\{P^1, P^0\}$. We say they are *complementary.* Bibel and Andrews' characterisation of validity for classical propositional logic is as follows:

THEOREM 1.1 (ANDREWS [AND81], BIBEL [BIB81]) *A propositional formula is classically valid if and only if every path through the (matrix representation of the) formula contains a complementary connection. That is, there exists a set of connections in the formula (matrix) that are said to* span *it.*

This characterisation translates the problem of checking a formula for validity (in classical logic) into a path-checking problem: *i.e.*, checking that every path through the formula contains (as a subpath) a connection. In the next section we consider how this path-checking can be performed.

1.4.2 Matrix-based proof procedures.

Both Bibel [Bib81,Bib82a,Bib82c], and Andrews [And81] develop proof procedures based on matrix characterisations of validity for classical logic. Bibel [Bib82b] shows that such procedures are comparable in efficiency with the standard refinements of resolution (and more efficient in most cases).

These proof procedures decompose into two components:

- a path-checking algorithm, and
- a method for testing the complementarity of two atomic formula occurrences.

For proof search in propositional logic, the second component is trivial; we need only test whether two atoms are the same proposition, and have different polarities. For first-order logic Robinson's unification algorithm [Rob65] can be utilised to test atoms for complementarity. The two atomic formulae of a connection must unify.

The crucial point is this: Bibel shows in [Bib82b] that the central component of such a proof procedure is the path-checking component. Provided the test for complementarity is computationally tractable, it can be called on demand as potential connections are identified. Each complementary connection serves to check *all* of the paths of which it is a subpath. The differences between many (resolution-based) proof procedures for classical logic can be analysed in terms of how many of the paths they successfully eliminate from consideration out of the set of paths that they are entitled to eliminate given a complementary connection. We refer the reader to Bibel's paper [Bib82b] for more details.

The other side of this coin is that real improvements in the efficiency of proof procedures are obtained by elaborating on this theme and trying to increase the number of paths eliminated by each connection. Bibel and his co-workers have developed many such efficient algorithms (see [Bib77,Bib82a,HB82] for example).

We describe a simple path-checking procedure below for concreteness. For a tutorial introduction to these methods the reader is referred to [Bib83].

The matrix representation of the example formula of the last section is:

$$\begin{array}{cccc} & P^0 & Q^0 & \\ \rightarrow P^1 & & & R^0 \\ & Q^1 & R^1 & \\ & \uparrow & & \end{array} \tag{1.1}$$

We choose a column (indicated by the vertical arrow) and an element of that column (indicated by the horizontal arrow). Next we search for a complementary atom in the remainder of the matrix (formula). We identify the occurrence of P^0 in the second column. This pair forms our first connection. We now eliminate all paths through the matrix that contain that connection. These paths are all those that pass through P^1 in the first column, continue into the second through P^0, and proceed onwards through the matrix. We move our vertical column marker to the second column and a horizontal marker to the second row of that column indicating that we are only interested in the paths that pass through the first column and continue through the second element of the second column. (We indicate the connection made with an arc.)

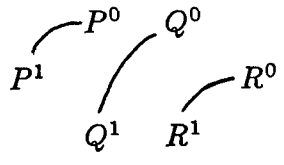
$$\begin{array}{cccc} & \overset{\frown}{P^0} & Q^0 & \\ \rightarrow P^1 & & & R^0 \\ & \rightarrow Q^1 & R^1 & \\ & \uparrow & & \end{array}$$

We now search for an atom in the remainder of the matrix that is complementary to Q^1. We find Q^0 (of course) and repeat the path elimination process outlined above. Our markers become:

$$\begin{array}{cccc} & \overset{\frown}{P^0} & \overset{\frown}{Q^0} & \\ \rightarrow P^1 & & & R^0 \\ & \rightarrow Q^1 & \rightarrow R^1 & \\ & & \uparrow & \end{array}$$

Notice how the horizontal markers keep a check on the set of paths we have yet to consider, namely those that pass through P^1 of the first column, Q^1 of the second column, R^1 of the third column, and continue into the rest of the matrix.

Repeating the process for R^1 we make the final connection and, since there are no paths left to check, the formula is proved valid by the three connections:



We refer the reader to Bibel's book for a more detailed account of such path-checking algorithms.

1.4.3 Path-checking algorithms for non-classical logics.

In the main body of this thesis we confine ourselves to a theoretical development of matrix characterisations of validity and do not develop any particular proof procedures based on these characterisations. We now justify this decision.

For each (non-classical) logic that we consider, we define what is meant by a "path" through a formula and how two atomic formulae may be complementary. These notions are thus logic-dependent. The validity of a formula is still captured in terms of the existence of a spanning set of connections, but with the notion of path and complementarity construed in this logic-dependent manner.

In fact, we go further. The logic-dependent component of the notion of a path does not affect the propositional part of the definition which gives us the basic matrix framework. In essence, the modal and intuitionistic matrix representation of a formula is the same as the classical definition. The distinction between, say, intuitionistic and classical logic, which share the same first-order language, is captured solely in terms of the definition of complementarity. Moreover, we show that the logic-dependent complementarity tests are tractable, for all of the non-classical logics considered, and can be implemented by the standard unification algorithm or minor modifications of existing equational unification algorithms.

Consequently, the path-checking algorithms developed for classical logic carry over *with no change* to the non-classical logics that we consider. For example, an implementation of a matrix-based proof procedure for S5 modal logic was

developed by making minor modifications to a matrix-based proof procedure for classical logic [WW87,Wil86].

Note that we are not saying that the complexity of checking a propositional formula valid in (propositional) intuitionistic or modal logic is the same as the complexity of checking the formula classically valid. The former problems are (excluding S5) PSPACE-complete, and the latter NP-complete. For classical propositional logic we need only consider one instance of the formula as our matrix, and check the paths through that. If we find a non-complementary path, we can conclude that the formula is falsifiable (and hence not valid). For the propositional fragments of both intuitionistic logic and the modal logics (excluding S5) we must consider additional "instances" of certain subformulae as well as the basic formula before we can conclude that the formula is falsifiable. This duplication increases the number of distinct paths through a formula and hence the number we have to check. The duplication is reminiscent of Herbrand's Theorem for first-order logic. To summarise: we can determine the validity of a formula in all of the logics with roughly the same computational resources, but, if we cannot immediately prove it valid, the space we must search in order to demonstrate its falsifiability is much larger in the intuitionistic and modal cases than in the classical case. This provides us with an interesting perspective on the rôle of complexity results in automated theorem proving.

1.4.4 Summary

In this section we have motivated our interest in extending matrix proof methods to non-classical logics. We have also outlined the properties of the classical matrix methods that we preserve in the non-classical matrix methods so as to render existing proof procedures applicable for the path-checking component of the search. We have also outlined a simple path-checking procedure for those readers not familiar with this work.

1.5 The structure of the thesis.

We have motivated our interest in efficient proof methods for non-classical logics (§1.2) and outlined the criteria we use to define the relative efficiency of proof procedures (§1.3). We have also indicated why we are interested in extending matrix proof methods (as opposed to other types of proof method) to non-classical logics and the properties of such methods that the extension must preserve to retain their computational properties (§1.4).

The thesis is divided into three parts, each dealing with proof search in a different logic or class of logics.

Part I: In this part of the thesis we are concerned with automated proof search in classical logic. The part contains two chapters.

Chapter 2: In this chapter we analyse and classify the redundancies in the search space induced by a standard sequent calculus for classical logic.

Chapter 3: In this chapter we present Bibel's Connection Calculus: a matrix characterisation of validity for classical logic, as a collection of theoretically motivated techniques for removing specific redundancies inherent in sequent-based proof search.

Part II: This part of the thesis is concerned with modal logics. It forms the main body of the thesis and comprises five chapters.

Chapter 4: In this chapter we present the syntax and semantics of the modal logics K, K4, D, D4, T, S4 and S5. We also present standard sequent calculi for a subset of these logics.

Chapter 5: In this chapter we analyse redundancies in the search space induced by the modal sequent calculi. Good use is made of the somewhat logic independent classification developed in Part I to classify

these redundancies and thus prepare the way for the application of the appropriate techniques for their removal.

Chapter 6: In this chapter we develop the matrix characterisations of validity for the target modal logics. We prove the characterisations correct and complete. This is the main theoretical chapter of the thesis.

Chapter 7: In this chapter we demonstrate properties of automated proof search based on the modal matrix characterisations. In particular, we demonstrate that the search spaces induced by the matrix characterisations are free from the redundancies identified within the sequent search space. We also discuss:

- The use of equational unification algorithms that form central components of proof procedures based on the modal matrix characterisations.
- The use of the matrix characterisations to develop efficient decision procedures for the propositional fragments of the target modal logics.

This is the main evaluation chapter in the thesis.

Chapter 8: In this chapter we discuss the relative merits of the other major proposals in the literature for the efficient automation of proof search in similar classes of modal logics. This is the main related work chapter.

Part III: This part of the thesis is concerned with first-order intuitionistic logic.

Chapter 9: In this chapter we repeat for the case of intuitionistic logic the development given in Part II for the modal logics. We present and analyse a standard sequent calculus for this logic, then proceed as before to develop a matrix characterisation of validity using the (by now) well-tried techniques identified in Part I.

Chapter 10 concludes the thesis with a summary and evaluation of the results obtained and a survey of interesting open problems, technical developments and alternative applications of the techniques developed and employed in this thesis.

Part I

Automated Proof Search in Classical Logic.

Summary.

In this initial part of the thesis we present a theoretical reconstruction of Bibel's Connection Calculus [Bib82a]: a matrix characterisation of validity for first-order classical logic. Our main contribution is not the final characterisation arrived at, which is essentially that given by Bibel in [Bib80,Bib82c], but the systematic method by which we develop it. That method is as follows:

- First, we analyse the proof search space induced by a standard, cut-free, sequent calculus for classical logic. We identify three problems of redundancy:
  - Notational redundancy: considerable duplication of the same information.
  - Relevance: the inclusion in the search space of branches that cannot lead to a proof.
  - Order dependence: the need to explore alternative branches in the search space that differ only in the order in which certain sequent rules are applied.
- Then we present the major features of the matrix characterisation, as principled methods for removing these redundancies.

In this way we view the matrix characterisation as a collection of theoretically motivated techniques for removing the redundancies inherent in sequent-based proof search, rather than as an adhoc proof method for classical logic (as it has been seen in the past). In short: we *derive*, or *generate*, the matrix characterisation from consideration of a cut-free sequent proof system [Wal86].

In Parts II and III we use the techniques identified in this part of the thesis to formulate matrix characterisations of validity for modal and intuitionistic logics respectively. We consider these results to be ample evidence of the power of the analysis presented here.

To reiterate: our contribution is not the formulation of the matrix characterisation for classical logic itself (which is essentially due to Bibel), but the identification of individual abstract techniques within Bibel's work for the removal of specific redundancies in sequent search spaces. We are then able to apply the techniques individually to other logics when and where they are applicable. As a particular case in point, consider our analysis of unification as a method for overcoming the order dependence of the sequent rules for quantifiers. The utility of unification for dealing with the substitutional aspects of classical quantifiers is well-known, but, in our opinion, the use to which it can be put in the automation of proof search in non-classical logics has not been apparent due to the traditional emphasis on Skolemisation within resolution-based systems. At best, it is seen as a technique for dealing with the quantificational aspects of such logics (see *eg.*, [AM86a,Kon86]). We show in Parts II and III that as a proof-theoretic tool for removing dependence on the application order of sequent rules unification can play a crucial rôle in the formulation of efficient proof procedures even for *propositional* modal and intuitionistic logics.

This part of the thesis comprises two chapters. In Chapter 2 we present a standard cut-free sequent calculus for classical logic and analyse the search space of derivations it induces. We identify the redundancies mentioned above by means of examples. In Chapter 3, taking each redundancy in turn, we introduce the appropriate theoretical structure to support its elimination from the search space. The final result is a matrix characterisation of validity for classical logic.

Chapter 2

Sequent-based proof search in classical logic.

2.1 Introduction.

In this chapter we present a standard cut-free sequent calculus for classical logic and analyse some properties of the search space induced by this inference system. Our attention is focussed on the suitability of the calculus as a basis for automated proof search.

First we review the language and semantics of classical logic for the sake of completeness (§2.2). Readers familiar with this material can safely skip this section. After that we introduce the sequent calculus and some terminology for the ensuing discussion (§2.3). In §2.4, we consider how the calculus can be used to search for proofs and describe the search space of derivations that it induces. Finally, in §2.5 we identify three problems of redundancy in this search space by means of examples. In Chapter 3 we formulate specific techniques for the removal of these redundancies. These techniques are abstractions of methods used by Bibel in the formulation of his Connection Calculus for classical logic [Bib80,Bib82c].

2.2 Syntax and semantics.

In this section we review the syntax and semantics of classical logic. This material is included for completeness and, since it is standard, the presentation is brief. Readers are referred to Smullyan's book [Smu68] for a more detailed development in the same style.

2.2.1 Syntax.

We consider a first-order language comprising:

1. A denumerable list of n-ary predicate symbols $P^n, Q^n, \ldots$, for each natural number n.

2. A denumerable list of individual variables x, y, z, (possibly subscripted).

3. An infinite set of individual constants c, d, (possibly subscripted).

4. The sentential connectives $\wedge$, $\vee$, $\Rightarrow$ and $\neg$.

5. The quantifiers $\forall$ and $\exists$.

As usual we assume the sets of symbols to be disjoint. We omit the arity of predicate symbols when it is clear from the context or irrelevant. We shall refer to the variables and constants together to be the *individual symbols*.

REMARKS. The above definition defines a class of languages dependent on particular choices of predicate, variable, and constant symbols. We assume some fixed set of predicate and variable symbols but allow the set of constants of the language to vary. If C is a set of constants, by "a first-order language over C" we mean a language defined as above whose constant symbols are among C.

Notice that the language contains no function symbols. We impose this restriction because it simplifies our discussions below. The matrix characterisations developed in the next chapter are therefore subject to this restriction. We

stress: the restriction is for technical convenience only. In a concluding section of the next chapter we show explicitly how this restriction can be lifted. (END OF REMARKS.)

Formulae can be defined inductively according to the following *formation* rules. The atomic formulae are $n+1$-tuples $Pc_1 \ldots c_n$, where P is an n-ary predicate symbol and the c_i, $i = 1, \ldots, n$, are individual symbols (*i.e.*, variables or constants). The set of formulae is the smallest set such that: if A and B are formulae, and x is an individual variable,

- the atomic formulae are formulae;
- $\neg A$, $A \wedge B$, $A \vee B$ and $A \Rightarrow B$ are formulae;
- $\forall x A$ and $\exists x A$ are formulae.

We shall use A, B and C as meta-variables ranging over formulae.

Notions such as "free" and "bound" occurrences of individual variables within a formula are defined as usual, as is the notion of the substitution of an individual constant c for a free variable x in A, denoted by $A[c/x]$. We shall not repeat the definitions explicitly here. Formulae with no free occurrences of variables are called *sentences*. *Pure* sentences are those containing no constants.

Immediate and *free immediate* subformulae are defined as follows:

- A is an immediate (and a free immediate) subformula of $\neg A$; both A and B are immediate (and free immediate) subformulae of $A \wedge B$, $A \vee B$ and $A \Rightarrow B$.
- For any individual constant c, individual variable x and formula A, $A[c/x]$ is an immediate subformula, and A is the free immediate subformula of $\forall x A$ and $\exists x A$.

The notion of a *(free) subformula* is defined by:

- If A is a (free) immediate subformula of B, or identical to B, A is a (free) subformula of B.

- If A is a (free) subformula of B, and B is a (free) subformula of C, A is a (free) subformula of C.

A *formation tree* for a formula A is a tree whose root is labelled by A, whose leaves are labelled by the free atomic subformulae of A, and whose interior nodes are formed as follows:

- If $\neg A$ is at an interior node, the node has one child with A at that node.

- If either $A \wedge B$, $A \vee B$ or $A \Rightarrow B$ is at an interior node, the node has two children with A at the first child and B at the second.

- If either $\forall x A$ or $\exists x A$ is at an interior node, the node has one child with A at that child.

Notice that the formation tree for a quantified formula is finitely branching and of finite depth.

REMARK. The notion of formation tree defined above is not quite standard. Formation trees are normally defined to be infinitely branching at nodes labelled by formulae whose major symbol is a quantifier, say $\forall x A$. The children of such a node are then labelled by the immediate subformulae of $\forall x A$, rather than its free immediate subformula as defined above. We find the definition given more suitable for our subsequent discussion. (END OF REMARK.)

2.2.2 Semantics.

We present a semantics for the first-order language(s) introduced in the last section based on that given by Smullyan [Smu68]. A similar form of semantics will be given for the modal logics in Part II.

Let D' be a set of constants, and D a non-empty set, called a *domain*, with $D \cap D' = \emptyset$. An *interpretation* (of the first-order language over D') in the domain

D is a mapping from D' into D, and from the n-ary predicates of the language to n-place relations on D. It is technically convenient to extend interpretations to $D' \cup D$, by defining them to be the identity function on D. A *model* is a pair $\langle D, \iota \rangle$ where D is a domain and ι an interpretation in that domain.

An atomic sentence $Pc_1 \cdots c_n$ is *true* in the model $\langle D, \iota \rangle$ just in case the n-tuple: $\iota(c_1), \ldots, \iota(c_n)$, is in the relation $\iota(P)$ over D^n. It is *false* in the model otherwise.

This defines the truth conditions for atomic sentences. The definition is extended to all sentences of the language as follows: for any model $\langle D, \iota \rangle$,

1. $\neg A$ is true in $\langle D, \iota \rangle$ iff A is false in $\langle D, \iota \rangle$.

2. $A \wedge B$ is true in $\langle D, \iota \rangle$ iff both A and B are true in $\langle D, \iota \rangle$.

3. $A \vee B$ is true in $\langle D, \iota \rangle$ iff either A or B is true in $\langle D, \iota \rangle$.

4. $A \Rightarrow B$ is true in $\langle D, \iota \rangle$ iff either A is false or B is true in $\langle D, \iota \rangle$.

5. $\forall x A$ is true in $\langle D, \iota \rangle$ iff for every $c \in D$, $A[c/x]$ is true in $\langle D, \iota \rangle$.

6. $\exists x A$ is true in $\langle D, \iota \rangle$ iff for some $c \in D$, $A[c/x]$ is true in $\langle D, \iota \rangle$.

A sentence is satisfiable if it is true in at least one model. A sentence is valid if it is true in all models.

2.3 The sequent calculus.

In this section we develop a cut-free sequent calculus for classical logic. The material is quite standard, so we introduce just the concepts and terminology utilised in the rest of this thesis. More leisurely treatments of the same material can be found in the books of Kleene [Kle68] or Gallier [Gal86], or Gentzen's original paper [G69].

We consider a first-order language as before, but with the set of constants comprising:

- A denumerable set of individual constants C_0.
- A denumerable set of parameters P_0.

Henceforth, we call the union of these sets D_0. We use a, b to denote parameters, and c, d to denote elements of D_0 (*i.e.*, individual constants or parameters).

REMARK. We use the calculus to prove sequents of the language over C_0 (*i.e.*, just including the individual constants) but the proof itself may involve sequents of the language over D_0 (*i.e.*, including both parameters and individual constants). We use parameters in the same way as some authors use free variables. (See *eg.*, [Kle68].) (END OF REMARK.)

A sequent is an ordered pair $\langle \Gamma, \Delta \rangle$ of finite sets of sentences over D_0, written $\Gamma \longrightarrow \Delta$. Γ is the *antecedent* and Δ the *succedent* of the sequent; we write $\longrightarrow \Delta$ and $\Gamma \longrightarrow$ for the sequents $\langle \emptyset, \Delta \rangle$ and $\langle \Gamma, \emptyset \rangle$ respectively. Following convention, we write Γ, A for the set $\Gamma \cup \{A\}$ and A, Δ for the set $\Delta \cup \{A\}$.

A sequent is interpreted semantically as follows: $\Gamma \longrightarrow \Delta$ is true in a model just in case, if all the elements of Γ are true in the model, at least one element of Δ is true in the model. A sequent is valid just in case it is valid in all models.

The rules of the calculus fall into three categories: *basic sequents* or axioms, *operational* rules and *structural* rules. Since our sequents are formed from sets

rather than sequences of formulae (in contrast to Gentzen's original formulation [G69]) we have no need for structural rules. This point is explained in more detail below. The basic sequents are instances of the schema:

$$\Gamma, A \longrightarrow A, \Delta .$$

The operational rules appear in pairs, each pair associated with a particular sentential connective or quantifier. One rule introduces the connective/quantifier into the antecedent, the other introduces it into the succedent. The operational rules for a sentential connective or quantifier permits the introduction of a formula with the symbol connective or quantifier as its major symbol into a sequent. There is one rule for introducing such a formula into antecedents, and one for introducing it into succedents. The complete system is summarised in Figure 2–1.

An occurrence of a formula of a sequent is called an S-formula (short for "sequent-formula"). If it appears in the antecedent of the sequent it may be called an *antecedent* S-formula; if it occurs in the succedent, it may be called a *succedent* S-formula.

In each of the operational rules, the sequent(s) above the line is called the *premise(s)*, and the sequent below the line the *conclusion* of the rule. The S-formula in the conclusion whose major symbol gives the rule its name, is called the *principal* S-formula of the inference. The occurrence(s) of its immediate subformula(e) in the premise(s) of the rule instance is (are) called the *side* S-formula(e) of the inference.

A *derivation* in this calculus is a tree structure in which each interior node is an instance of one of the rules given above (in the usual way). The sequent at the root of the tree is called the *endsequent* of the derivation, while the sequents at the leaves of the tree are called, appropriately, the *leaves* of the derivation. A derivation is said to be a derivation of its endsequent from its leaves. A *proof* of a sequent (over C_0) is a derivation of that sequent all of whose leaves are basic sequents.

$$\Gamma, A \longrightarrow A, \Delta$$

$$\frac{\Gamma \longrightarrow A, \Delta \quad \Gamma, B \longrightarrow \Delta}{\Gamma, A \Rightarrow B \longrightarrow \Delta} \Rightarrow\longrightarrow \qquad \frac{\Gamma, A \longrightarrow B, \Delta}{\Gamma \longrightarrow A \Rightarrow B, \Delta} \longrightarrow\Rightarrow$$

$$\frac{\Gamma, A, B \longrightarrow \Delta}{\Gamma, A \wedge B \longrightarrow \Delta} \wedge\longrightarrow \qquad \frac{\Gamma \longrightarrow A, \Delta \quad \Gamma \longrightarrow B, \Delta}{\Gamma \longrightarrow A \wedge B, \Delta} \longrightarrow\wedge$$

$$\frac{\Gamma, A \longrightarrow \Delta \quad \Gamma, B \longrightarrow \Delta}{\Gamma, A \vee B \longrightarrow \Delta} \vee\longrightarrow \qquad \frac{\Gamma \longrightarrow A, B, \Delta}{\Gamma \longrightarrow A \vee B, \Delta} \longrightarrow\vee$$

$$\frac{\Gamma \longrightarrow A, \Delta}{\Gamma, \neg A \longrightarrow \Delta} \neg\longrightarrow \qquad \frac{\Gamma, A \longrightarrow \Delta}{\Gamma \longrightarrow \neg A, \Delta} \longrightarrow\neg$$

$$\frac{\Gamma, A[c/x] \longrightarrow \Delta}{\Gamma, \forall x A \longrightarrow \Delta} \forall\longrightarrow \qquad \frac{\Gamma \longrightarrow A[a/x], \Delta}{\Gamma \longrightarrow \forall x A, \Delta} \longrightarrow\forall$$

$$\frac{\Gamma, A[a/x] \longrightarrow \Delta}{\Gamma, \exists x A \longrightarrow \Delta} \exists\longrightarrow \qquad \frac{\Gamma \longrightarrow A[c/x], \Delta}{\Gamma \longrightarrow \exists x A, \Delta} \longrightarrow\exists$$

For the $\longrightarrow \forall$ and $\exists \longrightarrow$ rules: the parameter a must not occur in the conclusion.

Figure 2–1: A cut-free sequent calculus for classical logic.

Thinning:

$$\frac{\Gamma \longrightarrow \Delta}{\Gamma,\, A \longrightarrow \Delta} \qquad \frac{\Gamma \longrightarrow \Delta}{\Gamma \longrightarrow A,\, \Delta}$$

Contraction:

$$\frac{\Gamma,\, A,\, A \longrightarrow \Delta}{\Gamma,\, A \longrightarrow \Delta} \qquad \frac{\Gamma \longrightarrow A,\, A,\, \Delta}{\Gamma \longrightarrow A,\, \Delta}$$

Interchange:

$$\frac{\Gamma,\, A,\, B \longrightarrow \Delta}{\Gamma,\, B,\, A \longrightarrow \Delta} \qquad \frac{\Gamma \longrightarrow A,\, B,\, \Delta}{\Gamma \longrightarrow B,\, A,\, \Delta}$$

Figure 2–2: Structural rules for Gentzen's sequent calculus.

In Gentzen's original formulation, the Γ and Δ in the sequent $\Gamma \longrightarrow \Delta$ stand for sequences of formulae. The calculus is then extended with rules for manipulating the structure of sequents. These rules permit:

- the introduction of new S-formulae into both the antecedent and the succedent;
- the contraction of two S-formulae that are identical as formulae in both antecedent and succedent, and
- the interchange of two S-formulae in both antecedent and succedent.

The rules, called "thinning," "contraction" and "interchange" rules respectively, are summarised in Figure 2–2.

In our formulation, the Γ and Δ in the sequent $\Gamma \longrightarrow \Delta$ stand for sets of formulae. Consequently,

$$\Gamma,\, A,\, A \;=\; \Gamma,\, A$$

and

$$\Gamma,\, A,\, B \;=\; \Gamma,\, B,\, A,$$

by the properties of sets alone. (Recall that we use Γ, A to denote $\Gamma \cup \{A\}$.) This explains the absence of contraction and interchange rules.

A further difference between Gentzen's formulation and ours occurs in the basic sequent. Gentzen took

$$A \longrightarrow A$$

as the schema for basic sequents. We have taken the more general schema:

$$\Gamma, A \longrightarrow A, \Delta .$$

Consequently there is no need in our formulation for the thinning rules. Formulae that would have been introduced by thinning can instead be introduced in the basic sequents. Our choice was based on the fact that we utilise the rules in an inverted fashion as reduction rules. Our formulation of the basic sequent is more natural for this method of proof search (see §2.4).

Set based formulations of the sequent calculus are quite common, *eg.*, [Smu68, Lyn66,Dum77], and we refer the reader to these authors for further details.

This concludes our presentation of the calculus. It is complete for classical logic in the sense that any sequent $\Gamma \longrightarrow \Delta$ (over C_0) is valid iff $\Gamma \longrightarrow \Delta$ is provable in the calculus (see *eg.*, [Kle68]). In particular, a sentence, A (over C_0), is valid iff the sequent: $\longrightarrow A$, is provable (by the definition of validity for sequents). In the next section we present a number of example derivations and proofs within this calculus.

REMARK. The calculus is called "cut-free" because of the absence of the so-called "cut" rule:

$$\frac{\Gamma \longrightarrow A, \Delta \qquad \Gamma, A \longrightarrow \Delta}{\Gamma \longrightarrow \Delta} \text{ cut}$$

This rule was included in Gentzen's original formulation of the sequent calculus to facilitate the translation of "natural deduction" proofs into proofs within the sequent calculus. Gentzen showed that any proof of a sequent involving the cut-rule could be translated into a proof in which the cut-rule is not used: a "cut-free" proof [G69]. (END OF REMARK.)

2.4 Search methods.

In this section we outline how the calculus may be utilised to determine the validity of a sentence, and hence how it induces a search space of derivations.

We have presented the sequent calculus as a set of axioms (basic sequents) and a set of inference rules (operational rules). To determine the validity of a sentence A (over C_0), we can start from some axioms and try to construct a proof of the sequent $\longrightarrow A$. A little thought will reveal that this *synthetic* method is not very directed. How are we to decide which basic sequents to start with? How can we decide which constructions are leading us toward the desired goal? We consider this method of proof search no further.

A more sensible method is to examine the sequent we wish to prove and see which rules could possibly be used to construct such a sequent. For example, suppose we are interested in determining the validity of the sentence: $(P \Rightarrow Q) \wedge (Q \Rightarrow R) \Rightarrow (P \Rightarrow R)$. We form the sequent:

$$\longrightarrow (P \Rightarrow Q) \wedge (Q \Rightarrow R) \Rightarrow (P \Rightarrow R),$$

and ask the question: "which rules have such a sequent as an instance of their conclusion?" Since there is only one S-formula, it must form the principal S-formula of the inference. Since it is a succedent S-formula, and its major connective is the implication symbol, the only rule with such a sequent as an instance of its conclusion is the $\longrightarrow\Rightarrow$ rule. We can thus form a derivation with the appropriate instance of the premise of the $\longrightarrow\Rightarrow$ rule as its only leaf, and our target sequent as endsequent:

$$\frac{(P \Rightarrow Q) \wedge (Q \Rightarrow R) \longrightarrow P \Rightarrow R}{\longrightarrow (P \Rightarrow Q) \wedge (Q \Rightarrow R) \Rightarrow (P \Rightarrow R)} \longrightarrow\Rightarrow \qquad (2.1)$$

Now repeat the process. Choose a leaf of derivation 2.1 (there is only one) and look at its structure. The sequent in question comprises two S-formulae. We are faced with a choice. A new derivation could be constructed by applying (in an inverted sense) the $\longrightarrow\Rightarrow$ rule once again, but this time with the current

succedent S-formula $P \Rightarrow R$ as principal formula. Alternatively, a new derivation could be constructed by applying (in an inverted sense) the rule $\wedge \longrightarrow$, with the antecedent S-formula $(P \Rightarrow Q) \wedge (Q \Rightarrow R)$ as principal S-formula. We show the derivations that result from each choice below.

$$\dfrac{\dfrac{(P \Rightarrow Q) \wedge (Q \Rightarrow R),\ P \longrightarrow R}{(P \Rightarrow Q) \wedge (Q \Rightarrow R) \longrightarrow P \Rightarrow R}\ \longrightarrow\Rightarrow}{\longrightarrow (P \Rightarrow Q) \wedge (Q \Rightarrow R) \Rightarrow (P \Rightarrow R)}\ \longrightarrow\Rightarrow \tag{2.2}$$

$$\dfrac{\dfrac{P \Rightarrow Q,\ Q \Rightarrow R \longrightarrow P \Rightarrow R}{(P \Rightarrow Q) \wedge (Q \Rightarrow R) \longrightarrow P \Rightarrow R}\ \wedge \longrightarrow}{\longrightarrow (P \Rightarrow Q) \wedge (Q \Rightarrow R) \Rightarrow (P \Rightarrow R)}\ \longrightarrow\Rightarrow \tag{2.3}$$

The choice we are faced with we shall term a *disjunctive* or "OR" choice. It is conceivable that by making the wrong choice at this point we will explore derivations that are not extendable to proofs.

REMARK. In actual fact, for the propositional fragment of this calculus, it can be shown that such a situation does not arise; *i.e.*, we can make this choice arbitrarily and are guaranteed not to rule out the possibility of extending the resultant derivation to a proof; that is if a proof of the endsequent exists at all. This is not the case for the full first-order calculus, nor for any of the sequent calculi we present for the "non-classical" logics considered in the sequel, even for their propositional fragments. We return to this issue in §2.5.3 below. (END OF REMARK.)

Continuing in this *analytic* manner, consider first derivation 2.2. It has only one leaf, in which there is only a single non-atomic S-formula: $(P \Rightarrow Q) \wedge (Q \Rightarrow R)$. The major symbol of this formula is the conjunction symbol. Consequently there is only one rule applicable: $\wedge \longrightarrow$. The resulting derivation is:

$$\dfrac{\dfrac{\dfrac{P \Rightarrow Q,\ Q \Rightarrow R,\ P \longrightarrow R}{(P \Rightarrow Q) \wedge (Q \Rightarrow R),\ P \longrightarrow R}\ \wedge \longrightarrow}{(P \Rightarrow Q) \wedge (Q \Rightarrow R) \longrightarrow P \Rightarrow R}\ \longrightarrow\Rightarrow}{\longrightarrow (P \Rightarrow Q) \wedge (Q \Rightarrow R) \Rightarrow (P \Rightarrow R)}\ \longrightarrow\Rightarrow \tag{2.4}$$

Now consider derivation 2.3. Although there is still only one leaf sequent, it contains three non-atomic S-formulae. All have the implication symbol as

their major symbol. We can apply the (inverted) rule $\longrightarrow\Rightarrow$ to $P \Rightarrow R$, or the (inverted) rule $\Rightarrow\longrightarrow$ to either of $P \Rightarrow Q$ or $Q \Rightarrow R$. We shall choose the first possibility. The resulting derivation is:

$$\dfrac{\dfrac{\dfrac{\dfrac{P \Rightarrow Q,\ Q \Rightarrow R,\ P \longrightarrow R}{P \Rightarrow Q,\ Q \Rightarrow R \longrightarrow P \Rightarrow R}\ \longrightarrow\Rightarrow}{(P \Rightarrow Q) \wedge (Q \Rightarrow R) \longrightarrow P \Rightarrow R}\ \wedge\longrightarrow}{\longrightarrow (P \Rightarrow Q) \wedge (Q \Rightarrow R) \Rightarrow (P \Rightarrow R)}\ \longrightarrow\Rightarrow \qquad (2.5)$$

Notice that the leaves of derivations 2.4 and 2.5 are identical even though the (inverted) rules were applied in different orders to construct them. Of course this would not have been the case had we made the last OR-choice differently. We shall return to this issue below (§2.5.3).

We abandon our parallel development and extend derivation 2.4 only. Choose the first antecedent S-formula of the leaf sequent of this derivation to form the principal S-formula of the next (inverted) rule application. (There is one other OR-choice at this point.) The $\Rightarrow\longrightarrow$ rule has two premises, so the derivation becomes:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Q \Rightarrow R,\ P \longrightarrow P, R \qquad Q \Rightarrow R, P, Q \longrightarrow R}{P \Rightarrow Q,\ Q \Rightarrow R,\ P \longrightarrow R}\ \Rightarrow\longrightarrow}{(P \Rightarrow Q) \wedge (Q \Rightarrow R),\ P \longrightarrow R}\ \wedge\longrightarrow}{(P \Rightarrow Q) \wedge (Q \Rightarrow R) \longrightarrow P \Rightarrow R}\ \longrightarrow\Rightarrow}{\longrightarrow (P \Rightarrow Q) \wedge (Q \Rightarrow R) \Rightarrow (P \Rightarrow R)}\ \longrightarrow\Rightarrow \qquad (2.6)$$

Notice that the left-hand leaf is a basic sequent since there is an occurrence of the atomic formula P in both the antecedent and succedent of the leaf. We need not extend the derivation at this leaf any further to produce a proof; the leaf is said to be *closed.* Of course, this state of affairs does not always pertain after the (inverted) application of a rule with two premises. If the leaf had not closed we would have been faced with a choice as to which of the two leaves of the derivation to extend next. However, since all of the leaves of a proof must close in order for the derivation to form a proof, we have at some time to extend all leaves that are not yet basic sequents. This sort of choice we call a *conjunctive* or "AND" choice, since we must consider both possibilities at some time.

Returning to the example. Derivation 2.6 has only one unclosed leaf, and that leaf contains only one non-atomic S-formula. We have no option but to apply the $\Rightarrow\longrightarrow$ rule once more. The resulting derivation is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P, Q \longrightarrow Q, R \qquad P, Q, R \longrightarrow R}{Q \Rightarrow R, P, Q \longrightarrow R} \Rightarrow\longrightarrow}{Q \Rightarrow R, P \longrightarrow P, R} \Rightarrow\longrightarrow}{P \Rightarrow Q, Q \Rightarrow R, P \longrightarrow R} \wedge\longrightarrow}{(P \Rightarrow Q) \wedge (Q \Rightarrow R), P \longrightarrow R} \longrightarrow\Rightarrow}{(P \Rightarrow Q) \wedge (Q \Rightarrow R) \longrightarrow P \Rightarrow R} \longrightarrow\Rightarrow}{\longrightarrow (P \Rightarrow Q) \wedge (Q \Rightarrow R) \Rightarrow (P \Rightarrow R)} \tag{2.7}$$

Notice that all leaves of this derivation are closed, hence it is a proof.

The discussion above indicates how the sequent calculus defines a search space of derivations. We say that the act of extending a derivation by applying an inverted sequent rule in the manner described *reduces* the principal S-formula of the inference. The leaf of the derivation is also said to have been reduced. The goal is to reduce the endsequent until all the leaves of the resulting derivation are closed. The result is the desired sequent proof. Used in this way the calculus is essentially an *analytic tableau* system (or more precisely, what Smullyan [Smu68] calls a *block* tableau system.) The inverted rules are the tableau reduction rules.

2.4.1 Duplication and generative S-formulae.

In our examples above we extended derivations with leaf sequents of the form:

$$A \wedge B \longrightarrow C$$

by reducing them as follows:

$$\dfrac{A, B \longrightarrow C}{A \wedge B \longrightarrow C} \wedge\longrightarrow \tag{2.8}$$

Because sequents are formed from sets, and

$$A \wedge B \longrightarrow C \quad = \quad A \wedge B, A \wedge B \longrightarrow C$$

the antecedent S-formula: $A \wedge B$, can be "preserved" through a reduction if we so desire. The reduction step would then look like this:

$$\dfrac{A \wedge B, A, B \longrightarrow C}{A \wedge B \longrightarrow C} \wedge\longrightarrow \tag{2.9}$$

Notice that we must choose between four S-formulae in order to extend derivation 2.9 (since there are four S-formula in the leaf of the derivation: $A \wedge B$, A, B and C), but between only three in order to extend derivation 2.8. We clearly desire to keep such duplication to a minimum while still retaining completeness.

The reader should check that we uniformly refused to duplicate formulae in this manner during reductions in our examples. It can easily be shown that for the propositional fragment of the calculus this decision does not compromise completeness. Unfortunately, for the full calculus this is not the case. The problem lies with the "universal" rules: $\forall \longrightarrow$ and $\longrightarrow \exists$ which, when used to reduce leaves of derivations, can introduce any constant or parameter into the leaf of the extended derivation.

In certain proofs, we must reduce:

- antecedent S-formulae of the form: $\forall x A$, or

- succedent S-formulae of the form: $\exists x A$,

more than once, each time with a different constant or parameter. We call S-formulae of this type *generative*. Duplication, in the manner described above, of generative S-formula is essential to retain completeness.

A simple example will illustrate the problem. Consider the sequent:

$$\longrightarrow \forall x(Px \vee Qx) \Rightarrow (Pc \vee Qd) \vee Pd$$

where c and d are individual constants. If we extend this derivation by reducing the implication and succedent disjunctions we reach:

$$\dfrac{\dfrac{\dfrac{\forall x(Px \vee Qx) \longrightarrow Pc,\ Qd,\ Pd}{\forall x(Px \vee Qx) \longrightarrow Pc \vee Qd,\ Pd}\ \longrightarrow \vee}{\forall x(Px \vee Qx) \longrightarrow Pc \vee Qd \vee Pd}\ \longrightarrow \vee}{\longrightarrow \forall x(Px \vee Qx) \Rightarrow (Pc \vee Qd) \vee Pd}\ \longrightarrow \Rightarrow \qquad (2.10)$$

There is only one S-formula to reduce in the leaf of derivation 2.10. We reduce this S-formula introducing the constant c. Without duplication, we get:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Pc \vee Qc \longrightarrow Pc,\ Qd,\ Pd}{\forall x(Px \vee Qx) \longrightarrow Pc,\ Qd,\ Pd}\ \forall \longrightarrow}{\forall x(Px \vee Qx) \longrightarrow Pc \vee Qd,\ Pd}\ \longrightarrow \vee}{\forall x(Px \vee Qx) \longrightarrow Pc \vee Qd \vee Pd}\ \longrightarrow \vee}{\longrightarrow \forall x(Px \vee Qx) \Rightarrow (Pc \vee Qd) \vee Pd}\ \longrightarrow \Rightarrow \qquad (2.11)$$

whereas with duplication we get:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\forall x(Px \vee Qx),\ Pc \vee Qc \longrightarrow Pc,\ Qd,\ Pd}{\forall x(Px \vee Qx) \longrightarrow Pc,\ Qd,\ Pd}\ \forall \longrightarrow}{\forall x(Px \vee Qx) \longrightarrow Pc \vee Qd,\ Pd}\ \longrightarrow \vee}{\forall x(Px \vee Qx) \longrightarrow Pc \vee Qd \vee Pd}\ \longrightarrow \vee}{\longrightarrow \forall x(Px \vee Qx) \Rightarrow (Pc \vee Qd) \vee Pd}\ \longrightarrow \Rightarrow \qquad (2.12)$$

Proceeding in parallel, there is only one possible extension of derivation 2.11 via the reduction of the disjunction:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Pc \longrightarrow Pc,\ Qd,\ Pd \qquad Qc \longrightarrow Pc,\ Qd,\ Pd}{Pc \vee Qc \longrightarrow Pc,\ Qd,\ Pd}\ \vee \longrightarrow}{\forall x(Px \vee Qx) \longrightarrow Pc,\ Qd,\ Pd}\ \forall \longrightarrow}{\forall x(Px \vee Qx) \longrightarrow Pc \vee Qd,\ Pd}\ \longrightarrow \vee}{\forall x(Px \vee Qx) \longrightarrow Pc \vee Qd \vee Pd}\ \longrightarrow \vee}{\longrightarrow \forall x(Px \vee Qx) \Rightarrow (Pc \vee Qd) \vee Pd}\ \longrightarrow \Rightarrow \qquad (2.13)$$

We choose the same reduction to extend derivation 2.12 giving us:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\forall x(Px \vee Qx),\ Pc \longrightarrow Pc,\ Qd,\ Pd \qquad \forall x(Px \vee Qx),\ Qc \longrightarrow Pc,\ Qd,\ Pd}{\forall x(Px \vee Qx),\ Pc \vee Qc \longrightarrow Pc,\ Qd,\ Pd}}{\forall x(Px \vee Qx) \longrightarrow Pc,\ Qd,\ Pd}}{\forall x(Px \vee Qx) \longrightarrow Pc \vee Qd,\ Pd}}{\forall x(Px \vee Qx) \longrightarrow Pc \vee Qd \vee Pd}}{\longrightarrow \forall x(Px \vee Qx) \Rightarrow (Pc \vee Qd) \vee Pd} \qquad (2.14)$$

The left-hand leaf of derivation 2.13 is closed, but the right-hand leaf is not. Furthermore, we cannot extend the derivation to a proof since all of the S-formulae of the right-hand leaf are atomic. The left-hand leaf of derivation 2.14 is also closed, and the right-hand leaf not. We can extend the derivation to a

proof by reducing the generative S-formula in the antecedent once again, this time introducing the parameter d. We leave the subsequent details to the reader.

We could formulate the calculus with alternative rules that reflect the need to consider such duplication of generative S-formulae. The alternative rules are as follows: (see *eg.*, [Kle68])

$$\frac{\Gamma, \forall x A, A[c/x] \longrightarrow \Delta}{\Gamma, \forall x A \longrightarrow \Delta} \; \forall \longrightarrow \qquad \frac{\Gamma \longrightarrow \exists x A, A[c/x], \Delta}{\Gamma \longrightarrow \exists x A, \Delta} \; \longrightarrow \exists$$

If we replace the corresponding rules of Figure 2–1 with the rules above, we can uniformly make the restriction that no duplication, other than that built-in to these quantifier rules, should occur during a reduction, and still retain completeness. With this change the system as used for proof search (*i.e.*, inverted) is precisely the block tableau of Smullyan [Smu68].

The method outlined above for determining the validity of a sentence is frequently proposed, and actually implemented, as a practical method for automated proof search; see *eg.*, [BT75,GMW79,D84,OS86]. Although such sequent-based methods are used for classical logic occasionally, they are of particular interest when the logic is non-classical, since sometimes they are the only proof methods available (*eg.*, [GMW79,Wri85]). Particular cases in point are modal and intuitionistic logics. In the next three sections we discuss certain properties of this search space by means of examples. Our conclusion is that sequent-based search is a poor method of determining the validity of sentences in classical and other logics. Our purpose in the rest of this thesis is to develop alternative methods.

2.5 Redundancy in the sequent search space.

In this section we investigate the structure of the search space induced by the sequent calculus. We identify three major redundancies in this space, which we term:

- notational redundancy (§2.5.1);

- relevance (§2.5.2), and
- order dependence (§2.5.3).

We present examples to illustrate each form of redundancy.

In Chapter 3 we derive a matrix characterisation of validity for classical logic based on the analysis contained in this section. In Parts II and III we perform a similar analysis of cut-free sequent calculi for modal and intuitionistic logics. These analyses also support the development of matrix characterisations of validity for these "non-classical" logics. These matrix characterisations avoid the above redundancies.

2.5.1 Notational redundancy.

The first issue we focus on is a simple one theoretically, but of great practical concern. It concerns the representation of the intermediate states of the search space induced by the sequent calculus. These states are derivations. If the formula being tested for validity is large, the intermediate derivations can themselves become very large. Moreover, there is a lot of shared structure between the derivations since formulae are repeated time and again within different sequents. Due to the existence of OR-choices in the search space, there is a need to maintain multiple states at one time (except perhaps if one adopts an incomplete depth-first search [CL73]). The representational overhead in implementing sequent-based proof search can become prohibitive.

The methods of analytic tableaux [Smu68,Fit83] overcome this problem to a certain extent, by not repeating formulae in distinct sequents, and maintaining information common to many sequents within one branch of the tableau. Even tableau, though, require the multiple representation of all the subformulae of a complex formula as it is progressively reduced.

Some of this repetition is however necessary. The reduction of a generative formula, say an antecedent S-formula of the form $\forall x A$, with the individual symbol c, introduces the subformula $A[c/x]$ to the derivation. As mentioned in the

previous section, we may need to consider multiple subformulae of $\forall x A$ in the derivation, *i.e.*, $A[c/x]$, $A[d/x]$, ... *etc*, to retain completeness.

In the next chapter we present a generalisation of Bibel's technique [Bib82c] for solving this problem and capturing the shared structure between sequents. The technique supports the representation of the subformulae of generative S-formulae, such as $\forall x A$ above, without recourse to copying the main structure of A. The technique is based on the notion of a *formula tree* and *multiplicity* and the technique of *structure sharing* [BM72] developed for resolution-based proof systems.

2.5.2 Relevance.

The second issue concerns the manner in which we make OR-choices during the search. Consider the (valid) formula: $A \wedge (B \wedge P) \Rightarrow P$, where P is an atomic formula and A and B are arbitrary formulae. The first two steps in the construction of a proof of this formula are completely determined, leading to the derivation:

$$\dfrac{\dfrac{A,\, B \wedge P \longrightarrow P}{A \wedge (B \wedge P) \longrightarrow P}\ \wedge \longrightarrow}{\longrightarrow A \wedge (B \wedge P) \Rightarrow P}\ \longrightarrow\Rightarrow$$

At this point we have an OR-choice: whether to reduce the S-formula A or the S-formula $B \wedge P$. Suppose our method of resolving OR-choices was to always reduce the "leftmost" S-formula in the sequent. Our next step would be to reduce A. If A were some complex formula we could expend a large amount of effort reducing it and its subformulae, even though such reductions may not help us reach a basic sequent. On the other hand, reduction of the other antecedent S-formula leads to a proof immediately. It should be clear that such pathological examples can be constructed whatever uniform method of resolving OR-choices we adopt. This is the problem we term *relevance*, and it is due to the fact that the obvious sequent-based search methods only utilise information concerning the major connectives of the S-formulae of the sequent being reduced [Wal86].

In the next chapter we introduce the notions of *path* and *connection* to overcome this problem. These notions are generalisations of those developed by Bibel [Bib81] and Andrews [And81]. The notion of "path" was considered originally for formulae in conjunctive normal-form by Prawitz [Pra60].

2.5.3 Order dependence.

We now come to the most fundamental problem with sequent-based proof search. The problems of notational redundancy and relevance are essentially propositional. The issue addressed in this section arises within the propositional fragment also, but is more serious within the full first-order system.

The search method outlined above consists of applying sequent rules in an inverted fashion to extend derivations. We say the leaf sequent is *reduced* by the rule application. We identified four sorts of choice in this process:

- AND-choices: which leaf of a derivation to extend.
- OR-choices: given a leaf, which S-formula of the sequent to reduce.
- Universal choices: if the S-formula to be reduced is generative, which parameter or constant to introduce.
- Existential choices: if the S-formula to be reduced is not generative, but requires the introduction of a parameter, which new parameter to choose.

Since every leaf of a derivation must be closed for it to constitute a proof, it is irrelevant how we make the AND-choices. Since for an existential choice, the parameter introduced must be new, we can choose arbitrarily from our denumerable set. We are left with the OR-choices and universal choices.

2.5.3.1 The propositional fragment.

Consider first the propositional fragment of the calculus, *i.e.*, there are no universal choices. We remarked in §2.4 that for the propositional fragment we can

make OR-choices arbitrarily; that is, fix a uniform method (such as "left-most" *etc*) of resolving the choice of which S-formula of a given leaf sequent to reduce. At worst, a bad choice will lead to a larger search as described in §2.5.2, but if the current leaf can be extended to a subtree all of whose leaves are closed, then any method of resolving OR-choices will eventually cause the subtree to close.

REMARK. We shall not formalise this result since it is well-known. An intuitive argument can be made as follows. Define the *degree* of a sequent to be the number of connectives in its S-formulae. Each operational rule of the propositional fragment of the calculus has the property that the degree of its premises is less than the degree of its conclusion. If we follow our restriction of never duplicating S-formulae when they are reduced, the degree of the new leaf sequent(s) of the extended derivation is less than the degree of the sequent from which they were generated. Any branch of a derivation (a path from the root of the derivation to a leaf) can be extended only a finite number of times before all the S-formula remaining in the leaf are atomic. If this sequent is not a basic sequent it forms the basis for a (propositional) model in which the succedent S-formula of the endsequent is false. Formalisations of this argument abound since it is a standard *systematic* method of proving completeness and decidability of classical propositional logic. (See for example, [Kle68,Smu68].) (END OF REMARK.)

One important consequence of this fact is that we can eliminate from the sequent search space the redundancy exemplified by derivations 2.4 and 2.5 of §2.4. Recall that these two derivations differed only in the order in which two S-formulae in the leaf sequent of derivation 2.1 were reduced. However both derivations have the same leaves, hence from the point of view of progress toward a basic sequent, both derivations are equivalent. If we can fix an order in which the S-formula of a given sequent are reduced, we can rule one or the other derivation out of the search space. This is similar to the selected literal restriction used in resolution-based systems [KK71].

In summary, we have argued that for the propositional fragment of the calculus the order in which the sequent rules are applied is not significant (except

for efficiency of course, as argued in §2.5.2). We say that there is no *order dependence* amongst the propositional rules.

2.5.3.2 The first-order case.

Consider now the full first-order system and the (valid) sentence: $\exists x \forall y P x y \Rightarrow \forall u \exists v P v u$. To search for a proof of this sentence we start with the sequent:

$$\longrightarrow \exists x \forall y P x y \Rightarrow \forall u \exists v P v u$$

The $\longrightarrow\Rightarrow$ rule is the only rule applicable to this sequent. Reduction gives us:

$$\frac{\exists x \forall y P x y \longrightarrow \forall u \exists v P v u}{\longrightarrow \exists x \forall y P x y \Rightarrow \forall u \exists v P v u} \quad \longrightarrow\Rightarrow \tag{2.15}$$

We are now faced with an OR-choice: whether to reduce the antecedent or succedent S-formula of the leaf. Either choice requires us to make an existential choice as to which parameter to introduce. We choose the former and introduce the parameter a. The extended derivation is:

$$\frac{\dfrac{\forall y P a y \longrightarrow \forall u \exists v P v u}{\exists x \forall y P x y \longrightarrow \forall u \exists v P v u}\ \exists \longrightarrow}{\longrightarrow \exists x \forall y P x y \Rightarrow \forall u \exists v P v u}\ \longrightarrow\Rightarrow \tag{2.16}$$

Notice that a does not appear in the leaf of derivation 2.15 hence the extension is correct.

Continuing from derivation 2.16, we are faced with another OR-choice: to reduce the antecedent or succedent S-formula of the leaf. We again choose to reduce the antecedent S-formula. This leads to a universal choice and we choose the parameter b for the reduction. The result is:

$$\frac{\dfrac{\dfrac{P a b \longrightarrow \forall u \exists v P v u}{\forall y P a y \longrightarrow \forall u \exists v P v u}\ \forall \longrightarrow}{\exists x \forall y P x y \longrightarrow \forall u \exists v P v u}\ \exists \longrightarrow}{\longrightarrow \exists x \forall y P x y \Rightarrow \forall u \exists v P v u}\ \longrightarrow\Rightarrow \tag{2.17}$$

We could have used the parameter a again, had we so desired, since the choice is a universal one.

We now come to the crucial issue. The leaf of derivation 2.17 contains only one non-atomic formula, so to extend the derivation we must apply the $\longrightarrow \forall$

rule. We are faced with an existential choice of parameter. That is, we must choose a parameter that does not occur in the current leaf. We cannot, therefore, introduce either a or b, but must choose a completely new parameter, say c. The result is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Pab \longrightarrow \exists v Pvc}{Pab \longrightarrow \forall u \exists v Pvu}\longrightarrow\forall}{\forall y Pay \longrightarrow \forall u \exists v Pvu}\forall\longrightarrow}{\exists x \forall y Pxy \longrightarrow \forall u \exists v Pvu}\exists\longrightarrow}{\longrightarrow \exists x \forall y Pxy \Rightarrow \forall u \exists v Pvu}\longrightarrow\Rightarrow \qquad (2.18)$$

Our aim in extending the derivation is to reach a basic sequent. Derivation 2.17 already contains one atomic formula in the antecedent, we need only construct a matching one in the succedent. However, any attempt to extend derivation 2.18 to a proof is doomed to failure since there is only one non-atomic S-formula: $\exists v Pvc$, and the antecedent atomic S-formula is not a subformula of it. That is, no reduction of $\exists v Pvc$ can yield an atomic formula of the form: Pab. The problem lies, of course, with the previous reduction that introduced the parameter c. However, in the reduction of $\forall u \exists v Pvu$, we were constrained to choose a new parameter and hence were unable to introduce the parameter b as required.

Recall that after the second reduction, resulting in derivation 2.16, we were faced with an OR-choice between reducing the antecedent formula: $\forall y Pay$, and the succedent formula: $\forall u \exists v Pvu$. We chose to reduce the former. Let us remake that choice. We extend derivation 2.16 by reducing the succedent formula of its leaf. We are faced with an existential choice, so we choose the new parameter b. The resultant derivation is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\forall y Pay \longrightarrow \exists v Pvb}{\forall y Pay \longrightarrow \forall u \exists v Pvu}\longrightarrow\forall}{\exists x \forall y Pxy \longrightarrow \forall u \exists v Pvu}\exists\longrightarrow}{\longrightarrow \exists x \forall y Pxy \Rightarrow \forall u \exists v Pvu}\longrightarrow\Rightarrow \qquad (2.19)$$

To extend this derivation we must choose again between the antecedent and succedent S-formulae. Both lead to universal parameter choices. We choose the former, and introduce the parameter b for the variable y so that the atomic side S-formula of the inference, Pab, is a subformula of the succedent S-formula:

$\exists v P v b$. The new derivation is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Pab \longrightarrow \exists v P v b}{\forall y P a y \longrightarrow \exists v P v b}\,\forall \longrightarrow}{\forall y P a y \longrightarrow \forall u \exists v P v u}\,\longrightarrow \forall}{\exists x \forall y P x y \longrightarrow \forall u \exists v P v u}\,\exists \longrightarrow}{\longrightarrow \exists x \forall y P x y \Rightarrow \forall u \exists v P v u}\,\longrightarrow\Rightarrow \qquad (2.20)$$

Finally we reduce the remaining non-atomic S-formula in the leaf sequent of derivation 2.20 introducing the parameter a so that the resulting atomic formula matches the antecedent atomic formula. We have constructed the proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Pab \longrightarrow Pab}{Pab \longrightarrow \exists v P v b}\,\longrightarrow \exists}{\forall y P a y \longrightarrow \exists v P v b}\,\forall \longrightarrow}{\forall y P a y \longrightarrow \forall u \exists v P v u}\,\longrightarrow \forall}{\exists x \forall y P x y \longrightarrow \forall u \exists v P v u}\,\exists \longrightarrow}{\longrightarrow \exists x \forall y P x y \Rightarrow \forall u \exists v P v u}\,\longrightarrow\Rightarrow \qquad (2.21)$$

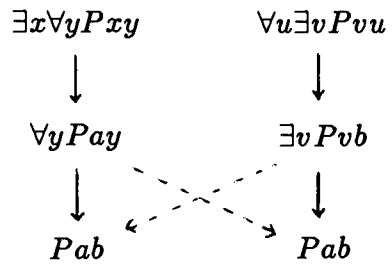
This example demonstrates that the order in which the quantifier rules are applied is significant. We cannot make OR-choices uniformly as in propositional logic. We say that there is an *order dependence* amongst the quantifier rules.

One way of exhibiting the constraints on the order in which the S-formulae may be reduced to yield a proof is to consider the order in which the immediate subformulae of quantified formulae are introduced as S-formulae. There are two subformulae of the S-formula $\exists x \forall y P x y$ that appear in the derivation as S-formulae: $\forall y P a y$ and Pab. The former must be introduced as an S-formula before the latter, since the latter is a subformula of the former. Similar constraints hold between the two subformulae, $\exists v P v b$ and Pab, of $\forall u \exists v P v u$. This gives us the following constraints on the introduction of subformulae of the endsequent of our example:

$$\begin{array}{ccc} \exists x \forall y P x y & & \forall u \exists v P v u \\ \downarrow & & \downarrow \\ \forall y P a y & & \exists v P v b \\ \downarrow & & \downarrow \\ Pab & & Pab \end{array}$$

where a solid arrow from a formula A to B indicates that B is an immediate subformula of A, and hence B will become an S-formula after A.

Now, because we desire the two atomic subformulae of the endsequent to form the pair of distinguished S-formulae in a basic sequent, we require that the same parameter, say a, be substituted for the pair x and v, and the same parameter, say b, substituted for the pair y and u. Since $\forall u \exists v P v u$ is not a generative S-formula in the derivation (it is a succedent S-formula of the form $\forall u A$) the choice of parameter for u is an existential one and subject to the restriction that the parameter be new. If we wish to substitute b for u, we must ensure that b does not already occur in the sequent. The only way that b could occur in the leaf sequent is if the reduction of $\forall y P a y$ to introduce Pab has already occurred. This is exactly what went wrong when we formed derivation 2.17. Therefore we infer a further constraint for our diagram, namely that the introduction of the subformula Pab of $\forall y P a y$ should occur *after* the introduction of $\exists v P v b$. A similar constraint holds between the subformulae Pab (of $\exists v P v b$) and $\forall y P a y$. We represent these constraints as additional dotted arrows in the diagram below.



Again, a dotted arrow from A to B indicates that B must become an S-formulae after A. The diagram, called a *reduction ordering* in the sequel, represents the orders in which S-formula in the derivation must be introduced in order to construct a basic sequent from instances of the free atomic subformulae of the endsequent. Notice that the order chosen at first above violates the constraints in that $\forall y P a y$ was reduced to Pab before the reduction of $\forall u \exists v P v u$ introduced $\exists v P v b$. In our second attempt the subformulae were introduced respecting the constraints in the diagram above, and a proof resulted.

In summary, we cannot adopt a uniform method of resolving OR-choices as was the case for the propositional fragment of the calculus. There is an order dependence amongst the quantifier rules. This is a fundamental problem with sequent-based search, since it means that the search space must contain

derivations that differ solely in the order in which certain rules are applied. Some orders may lead to a proof, some may not. The crucial point is that in order to traverse the sequent search space we must actually make a choice as to the reduction order. We have shown above that a sequent-based search strategy must be prepared to remake that choice.

In the next chapter we show how unification can be used to overcome this order dependency by removing the need actually to choose a given reduction order as the search proceeds. Instead only the *existence* of at least one correct reduction order that respects the caveats on the existential rules is checked.

2.6 Summary.

In this chapter we have:

- briefly reviewed the syntax and semantics of classical logic;
- introduced a cut-free sequent calculus and shown how it induces a proof search space, and
- identified three forms of redundancy within that search space which we have termed *notational, relevance* and *order dependence.*

In the next chapter we develop specific techniques for the removal of these redundancies. The techniques are abstractions of methods used by Bibel in the formulation of his Connection Calculus for classical logic [Bib80,Bib82c]. The particular techniques (paths and connections) for the removal of problems concerning relevance were also developed independently by Andrews [And81]. The combination of these techniques results in a matrix characterisation of validity for classical logic which is free from the specific problems of sequent-based search identified above. The matrix method therefore forms a more suitable basis for automated proof search in classical logic.

Chapter 3

A matrix characterisation of validity in classical logic.

3.1 Introduction.

In the previous chapter we presented a standard cut-free sequent calculus for classical logic and demonstrated one way in which it induces a search space of derivations. We analysed this search space and identified three types of redundancy within it. The redundancies were as follows:

- Notational redundancy: considerable duplication of the same information.
- Relevance: the inclusion in the search space of branches that cannot lead to a proof.
- Order dependence: the need to explore alternative branches in the search space that differ only in the order in which certain sequent rules are applied.

In this chapter we consider each problem in turn and introduce appropriate theoretical structure to alleviate it within practical proof procedures. The final result is a matrix characterisation of validity for classical logic free from the aforementioned redundancies. The central arguments of this analysis were first published in [Wal86].

The matrix characterisation so derived is, in essence, the Connection Calculus of Bibel [Bib82a,Bib82c]. We make no claim for originality in the basic nature of the characterisation. Our contribution is in the decomposition of the Connection Calculus into a set of theoretically based techniques for overcoming redundancies arising in sequent-based proof search. The significance of this contribution is that, once isolated, these techniques can be applied individually to other logics in novel ways to improve the efficiency of automated proof search in those logics. We do this to good effect in Parts II and III of this thesis where we develop matrix characterisations of validity for a wide range of non-classical logics.

This chapter is divided into four major sections. In §3.2 we introduce techniques for removing the notational redundancies of the sequent search space based on the notions of *formula trees* and *positions*. The techniques amount to a theoretically motivated version of "structure sharing" [BM72]: a technique originally developed to deal with similar forms of notational redundancy arising within resolution-based proof systems. We show that the applicability of the technique relies on the *subformula property* possessed by the cut-free sequent calculus.

In §3.3 we deal with the problem of relevance via the notions of *path*, *polarity* and *connection*. It is the notion of path that makes the label "matrix" appropriate for the final characterisation and gives proof methods based on the characterisation their path-checking flavour. A path through a formula, A, is defined so as to represent a potential leaf of a sequent derivation of the endsequent: $\longrightarrow A$. The set of paths through A represents the set of potential leaves of *any* sequent derivation of $\longrightarrow A$ (given a bound on the number of duplications of subformulae of A). A connection in the formula is a representation of the distinguished atomic S-formulae within a (potential) basic sequent. It consists of two atomic formula occurrences with the same predicate symbol and of differing polarities (*i.e.*, a positive and a negative occurrence of a given proposition). If a path contains a connection it has the potential to represent a closed leaf of a derivation. To overcome problems of relevance we search for connections directly rather than adopt the standard connective-based approach. That is, we

search directly amongst the potentially closed leaves for a subset suitable for the formation of a proof of the endsequent.

The effect of this is to replace an indirect search for basic sequents with a directed one. This eliminates from the (direct) matrix search space those parts of the (indirect) sequent search space that are irrelevant for the construction of basic sequents and hence proofs. If every path through A contains one of a given set of connections the set is said to *span* the formula. For propositional logic we have no need to consider the internal structure of derivations at all. Consequently the existence of a spanning set of connections for A entails the existence of a sequent proof of $\longrightarrow A$ and therefore the classical validity of A (and vice versa).

For first-order logic however the internal structure of derivations is important since there are constraints imposed on the application of certain quantifier rules: the problem of "order dependence." In §3.4 we introduce a mapping that represents the coherence of the choice of parameters for the free variables of (free) atomic subformulae in a derivation. Such coherence is required so that the two atomic components of a connection can still be construed as the distinguished antecedent and succedent formulae of a basic sequent (*i.e.*, they must be identical as formulae). Any such mapping induces a *reduction ordering*: a transitive relation over subformulae of the endsequent that we are proving. The reduction ordering represents the constraints on the order in which immediate subformulae of quantified (sub)formulae may be introduced as S-formulae into a derivation. The constraints arise from the provisos on two existential rules: $\longrightarrow \forall$ and $\exists \longrightarrow$. *Admissible* mappings are those whose reduction orderings are irreflexive.

A connection is defined to be *complementary* under an admissible mapping just in case its atomic components are identical under the mapping. The admissibility condition then ensures that at least one sequent derivation exists in which the required coherence in the choice of parameters is realised so that the current set of connections form the closed leaves of the derivation. A spanning set of connections, complementary under some such admissible mapping, thus entails

the existence of a sequent proof of $\longrightarrow A$, and hence the (first-order) validity of the formula (and vice versa). Robinson's unification algorithm (or more efficient refinements of it) can be used to compute the appropriate mappings. Unification is used to ensure the *existence* of a correct order of sequent rule applications to produce a proof of the formula. No single concrete order need be preferred. This technique removes the order dependence, induced in the sequent search space by the quantifier rules, from the matrix search space.

Whilst the basic nature of the characterisation is due to Bibel, we believe our particular *technical* formulation of the individual techniques to be quite significant. For example, we utilise Smullyan's uniform notation [Smu68] extensively and reformulate the notion of path to relate it more closely with sequent-based ideas. We shall remark on the technical differences between our formulation and that of Bibel and Andrews as we go. In §3.5 we summarise the relationships and argue the case for our formulation which make the individual techniques comprising the matrix methods more widely accessible and applicable.

In the remainder of this introductary section we introduce Smullyan's [Smu68] uniform notation.

3.1.1 Uniform notation.

In this preliminary section we introduce Smullyan's uniform notation for formulae over a first-order language. This notation, and others notations based on it, are utilised extensively in this thesis.

A *signed* formula is a pair $\langle A, n\rangle$ where A is a formula (over D) and $n \in \{0, 1\}$. We let X, Y, Z, possibly subscripted, range over signed formulae. A signed formula is said to be atomic if its constituent formula is atomic; otherwise it is non-atomic.

Non-atomic signed formulae are classified as follows:

1. A signed formula of the form: $\langle A \wedge B, 1\rangle$, $\langle A \vee B, 0\rangle$, $\langle A \Rightarrow B, 0\rangle$, $\langle \neg A, 1\rangle$ or $\langle \neg A, 0\rangle$ is of *conjunctive* or α-type. We shall sometimes use the symbol

"α" to stand for a formula of this form. We define the *components*, α_1 and α_2 of such a formula as follows:

- If $\alpha = \langle A \wedge B, 1\rangle$, then $\alpha_1 = \langle A, 1\rangle$ and $\alpha_2 = \langle B, 1\rangle$.
- If $\alpha = \langle A \vee B, 0\rangle$, then $\alpha_1 = \langle A, 0\rangle$ and $\alpha_2 = \langle B, 0\rangle$.
- If $\alpha = \langle A \Rightarrow B, 0\rangle$, then $\alpha_1 = \langle A, 1\rangle$ and $\alpha_2 = \langle B, 0\rangle$.
- If $\alpha = \langle \neg A, 1\rangle$, then $\alpha_1 = \langle A, 0\rangle$ and $\alpha_2 = \langle A, 0\rangle$.

2. A signed formula of the form: $\langle A \wedge B, 0\rangle$, $\langle A \vee B, 1\rangle$, $\langle A \Rightarrow B, 1\rangle$ is of *disjunctive* or β-type. We shall sometimes use the symbol "β" to stand for a formula of this form. We define the components, β_1 and β_2, of such a formula as follows:

   - If $\beta = \langle A \wedge B, 0\rangle$, then $\beta_1 = \langle A, 0\rangle$ and $\beta_2 = \langle B, 0\rangle$.
   - If $\beta = \langle A \vee B, 1\rangle$, then $\beta_1 = \langle A, 1\rangle$ and $\beta_2 = \langle B, 1\rangle$.
   - If $\beta = \langle A \Rightarrow B, 1\rangle$, then $\beta_1 = \langle A, 0\rangle$ and $\beta_2 = \langle B, 1\rangle$.

3. A signed formula of the form: $\langle \forall x A, 1\rangle$ or $\langle \exists x A, 0\rangle$ is of *universal* or γ-type. We shall sometimes use the symbol "γ" to stand for a formula of this form. The components, $\gamma_0(c)$, for $c \in D$, of such a formula are defined as follows: for $c \in D$,

   - If $\gamma = \langle \forall x A, 1\rangle$, then $\gamma_0(c) = \langle A[c/x], 1\rangle$.
   - If $\gamma = \langle \exists x A, 0\rangle$, then $\gamma_0(c) = \langle A[c/x], 0\rangle$.

4. A signed formula of the form: $\langle \forall x A, 0\rangle$ or $\langle \exists x A, 1\rangle$ is of *existential* or δ-type. We shall sometimes use the symbol "δ" to stand for a formula of this form. The components, $\delta(c)$, for $c \in D$, of such a formula are defined as follows: for $c \in D$,

   - If $\delta = \langle \forall x A, 0\rangle$, then $\delta_0(c) = \langle A[c/x], 0\rangle$.
   - If $\delta = \langle \exists x A, 1\rangle$, then $\delta_0(c) = \langle A[c/x], 1\rangle$.

α	α_1	α_2
$\langle A \wedge B, 1\rangle$	$\langle A, 1\rangle$	$\langle B, 1\rangle$
$\langle A \vee B, 0\rangle$	$\langle A, 0\rangle$	$\langle B, 0\rangle$
$\langle A \Rightarrow B, 0\rangle$	$\langle A, 1\rangle$	$\langle B, 0\rangle$
$\langle \neg A, 1\rangle$	$\langle A, 0\rangle$	$\langle A, 0\rangle$
$\langle \neg A, 0\rangle$	$\langle A, 1\rangle$	$\langle A, 1\rangle$

γ	$\gamma_0(a)$
$\langle \forall x A, 1\rangle$	$\langle A[a/x], 1\rangle$
$\langle \exists x A, 0\rangle$	$\langle A[a/x], 0\rangle$

β	β_1	β_2
$\langle A \wedge B, 0\rangle$	$\langle A, 0\rangle$	$\langle B, 0\rangle$
$\langle A \vee B, 1\rangle$	$\langle A, 1\rangle$	$\langle B, 1\rangle$
$\langle A \Rightarrow B, 1\rangle$	$\langle A, 0\rangle$	$\langle B, 1\rangle$

δ	$\delta_0(a)$
$\langle \forall x A, 0\rangle$	$\langle A[a/x], 0\rangle$
$\langle \exists x A, 1\rangle$	$\langle A[a/x], 1\rangle$

Table 3–1: Uniform notation for signed formulae.

Table 3–1 summarises the complete classification.

3.2 Formula trees and notational redundancy.

Gentzen's original motivation for defining sequent calculi was that such systems were more amenable to certain meta-mathematical arguments he had in mind than the calculi of "natural deduction" in which he was primarily interested. In particular the cut-free sequent systems possess what he called a *subformula property.* Simply stated this amounts to the fact that the S-formulae occurring in any derivation of a given endsequent are all subformulae of that endsequent. Since his original paper this property has become the defining characteristic of an "acceptable" cut-free sequent calculus for any logic (see [Sat77] for instance).

A study of the individual rules of our formulation of the sequent calculus, presented in Figure 2–1 of the last chapter, reveals that this formulation also

has the subformula property. Notice that the premise(s) of each rule is formed completely from subformulae of the conclusion of that rule. This is the proof-theoretic basis for the solution to notational redundancy embedded in Bibel's Connection Calculus, though he does not explicitly make this association. We shall encode a derivation in terms of the subformulae of its endsequent. From a theoretical standpoint, we use a structure called a *formula tree* which contains a name, or *position*, for each subformula of the endsequent. From a practical standpoint, positions are understood as pointers into a single concrete representation of the endsequent in the database of the proof procedure. During the search all intermediate derivations can be encoded in terms of these pointers. We now present the details.

3.2.1 Formula trees for formulae.

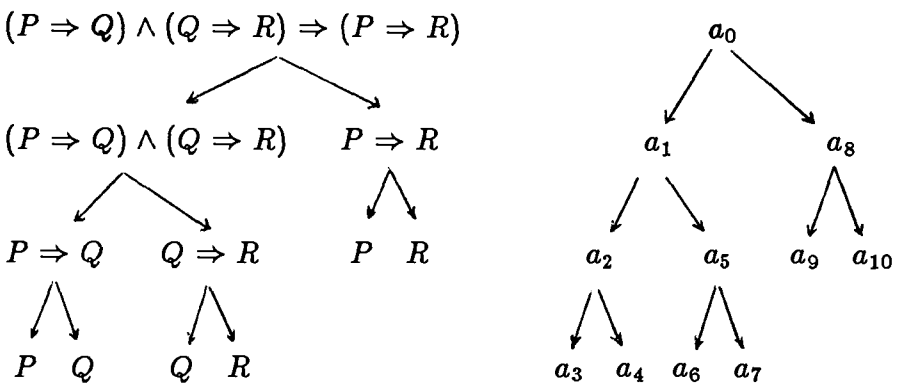
A formula tree for a sentence is a representation of its formation tree. It is best explained by example. We include in the formula tree a node, or *position*, for each free subformula of the sentence. The tree ordering is then defined as the subformula ordering: a position k is above a position k' in the tree, written $k \ll k'$, provided the formula associated with k' is a proper (free) subformula of the formula associated with k. A formula tree for the formula:

$$(P \Rightarrow Q) \wedge (Q \Rightarrow R) \Rightarrow (P \Rightarrow R)$$

is shown in Figure 3–1. We shall use k, l as meta-variables to range over positions of formula trees, and $\ll$ to denote the tree ordering over such positions. For a position k of a formula tree we use $\mathrm{lab}(k)$ to denote the subformula named by or associated with k. A position labelled by an atomic formula is called an *atomic position*. The labels of the positions for our example are also shown in the figure.

3.2.2 Formula trees for signed formulae.

The positions of a formula tree for a formula A are not quite rich enough nor flexible enough to identify the formulae that make up an arbitrary derivation of



k	lab(k)
a_0	$(P \Rightarrow Q) \wedge (Q \Rightarrow R) \Rightarrow (P \Rightarrow R)$
a_1	$(P \Rightarrow Q) \wedge (Q \Rightarrow R)$
a_2	$P \Rightarrow Q$
a_3	P
a_4	Q
a_5	$Q \Rightarrow R$
a_6	Q
a_7	R
a_8	$P \Rightarrow R$
a_9	P
a_{10}	R

Figure 3–1: Example formation and formula tree for a formula.

the endsequent $\longrightarrow A$. The first problem is that occurrences of formulae in a derivation occur either in antecedents or succedents (*i.e.*, they are S-formulae). We must represent this component of a formula occurrence also. The second problem concerns quantifiers and will be dealt with in §3.2.3.

Luckily, another proof-theoretic property of sequent systems supports a succinct solution to the first problem. First we need some terminology. Let B be a particular subformula of A. We can trace the progress of this subformula through a derivation until it becomes an S-formula of some sequent within the derivation (if it ever does). We call all these occurrences of B the *images* of B in the derivation. An example will help. In the derivation below, taken from the previous chapter, we have "boxed" the images of the first occurrence of the atomic formula P in the endsequent.

$$\dfrac{\dfrac{\dfrac{\dfrac{Q \Rightarrow R,\ P \longrightarrow \boxed{\mathrm{P}},\ R \qquad \dfrac{P,\ Q \longrightarrow Q,\ R \qquad P,\ Q,\ R \longrightarrow R}{Q \Rightarrow R,\ P,\ Q \longrightarrow R}}{\boxed{\mathrm{P}} \Rightarrow Q,\ Q \Rightarrow R,\ P \longrightarrow R}}{(\boxed{\mathrm{P}} \Rightarrow Q) \wedge (Q \Rightarrow R),\ P \longrightarrow R}}{(\boxed{\mathrm{P}} \Rightarrow Q) \wedge (Q \Rightarrow R) \longrightarrow P \Rightarrow R}}{\longrightarrow (\boxed{\mathrm{P}} \Rightarrow Q) \wedge (Q \Rightarrow R) \Rightarrow (P \Rightarrow R)}$$

A subformula B (free or otherwise) of a given formula A is said to occur positively in A if it occurs within an even number of explicit or implicit negations; otherwise it occurs negatively in A. An explicit negation is, of course, the negation symbol. Implicit negations arise via implications (and sequent arrows). Formally,

- B is a positive subformula of B.
- If a distinguished occurrence of B in A occurs positively (negatively) within A, then it occurs positively (negatively) within: $A \wedge C$, $C \wedge A$, $A \vee C$, $C \vee A$, $C \Rightarrow A$, $\forall x A$, $\exists x A$ and $\Gamma,\ A \longrightarrow \Delta$; and negatively (positively) within: $\neg A$, $A \Rightarrow C$ and $\Gamma \longrightarrow A,\ \Delta$.

Notice that we have extended the definition to sequents as well.

The well-known proof-theoretic property of sequent calculi referred to above is that an image of a subformula of the endsequent can occur as an antecedent

(succedent) S-formula if and only if it is a positive (negative) subformula of the endsequent. The basic property of sequent calculi that preserves this invariant is that only side formulae occurring negatively within the principal formula of an inference change sides from antecedent to succedent or vice versa. The reader can check that this is the case in our formulation of the calculus in Figure 2–1. The rules to concentrate on are the implication and negation rules.

Consequently we can identify whether images of a subformula of the endsequent will appear as an antecedent or succedent S-formula within derivations of that endsequent by noting whether the subformula occurs positively or negatively within the endsequent. For example, the distinguished occurrence of P occurs positively in the formula:

$$(P \Rightarrow Q) \wedge (Q \Rightarrow R) \Rightarrow (P \Rightarrow R),$$

but negatively within the sequent:

$$\longrightarrow (P \Rightarrow Q) \wedge (Q \Rightarrow R) \Rightarrow (P \Rightarrow R).$$

(There are three implicit negations: two implications and one sequent arrow). Consequently the images of this distinguished occurrence of P appear as succedent S-formulae in a derivation of the endsequent (if they appear as S-formulae at all).

To capture these notions in terms of formula trees we define a formula tree for a *signed formula*. Recall that a signed formula is a pair, $\langle A, n \rangle$, where A is a formula and $n \in \{0, 1\}$. A signed formula can be used to represent an S-formula as follows:

- $\langle A, 0 \rangle$ represents a succedent S-formula A.
- $\langle A, 1 \rangle$ represents an antecedent S-formula A.

In particular, we are interested in formula trees for signed formulae of the form: $\langle A, 0 \rangle$, since this signed formula represents the sequent: $\longrightarrow A$.

Formally, a formula tree for a signed formula, $\langle A, n\rangle$, is a formula tree for the formula A together with an assignment of a polarity, $\mathrm{pol}(k)$, to each position, k, of the formula tree. Polarities are assigned as follows:

- If $\mathrm{lab}(k)$ occurs positively within A, then $\mathrm{pol}(k) = n$.
- If $\mathrm{lab}(k)$ occurs negatively within A, then $\mathrm{pol}(k) = (n+1) \bmod 1$.

A formula tree for the signed formula:

$$\langle (P \Rightarrow Q) \wedge (Q \Rightarrow R) \Rightarrow (P \Rightarrow R), 0\rangle$$

is the formula tree shown in Figure 3–1 together with the polarity assignment:

k	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
$\mathrm{pol}(k)$	0	1	1	0	1	1	0	1	0	1	0

Each position, k, of the formula tree for a signed formula itself denotes a signed formula, $\mathrm{sform}(k)$, as follows:

$$\mathrm{sform}(k) \stackrel{\mathrm{df}}{=} \langle \mathrm{lab}(k), \mathrm{pol}(k)\rangle.$$

We summarise our progress up to now. Our goal is to represent derivations of an endsequent, say $\longrightarrow A$, in terms of the subformulae of A. We have introduced names, called "positions," for each distinct free subformula of A and arranged them in a tree reflecting the subformulae ordering in A. The subformula associated with a position k is its "label" denoted by $\mathrm{lab}(k)$. In order to represent the S-formulae of derivations using positions we associate a "polarity," $\mathrm{pol}(k)$, indicating whether images of the label of k appear as succedent ($\mathrm{pol}(k) = 0$) or antecedent ($\mathrm{pol}(k) = 1$) S-formulae. This information is obtained by noting whether $\mathrm{lab}(k)$ occurs positively or negatively within the endsequent.

REMARKS. Bibel's [Bib80,Bib82c] formulation of a formula tree corresponds to our notion of a formula tree for a *formula*. He does not consider formula trees for signed formulae explicitly. Instead, he makes an initial restriction to formulae in *negation normal-form*: a normal form in which negation symbols

only dominate atomic formulae and there are no implication signs. Under this restriction only the polarity of atomic formulae need be considered and the notion of a formula tree for a formula suffices to encode derivations. We note that this inessential restriction has led to confusion in the literature [Mur82].

Our formulation demonstrates how to lift this restriction and is therefore more general. This abstraction from specific properties of classical logic is important to permit the use of the technique in logics which do not admit a negation normal form. Intuitionistic logic is one such logic. Due to the more general formulation of the technique presented in this section we do not encounter any problems in utilising it to improve proof search in that logic. The details are presented in Part III.

We note that Andrews [And81] does not concern himself with notational redundancy. His systems do not include any techniques similar to those presented by Bibel in [Bib82a,Bib82c] and generalised here. (END OF REMARKS.)

3.2.3 Multiplicities and indexed formula trees.

The notions we have introduced so far are sufficient to encode derivations in the propositional fragment of the calculus but not in the full first-order system. The problem is that we can only represent free subformulae of the endsequent by positions. For propositional logic, this suffices since the notions of "free subformula" and "subformula" coincide. For first-order formulae we need to distinguish different instances of subformulae formed by the substitution of distinct parameters for the free variables of free subformulae of the endsequent.

As a first step we identify the positions of a formula tree that represent generative subformulae of the endsequent. Recall that a position k of a formula tree for the signed formula $\langle A, 0\rangle$ represents a signed formula sform(k). We give to each non-atomic position k a "type" depending on the type of sform(k) in Smullyan's classification presented in the introduction to this chapter. We call this the *principal* type of the position and denote it by Ptype(k). There are four principal types: α, β, γ and δ.

Each position, except for the root position, has a parent in the formula tree. Smullyan's classification gives us names for the components (immediate subformulae) of signed formulae as well as for formulae themselves. We give to each non-atomic position k, a *secondary* type determined by considering sform(k) as a component of the signed formula represented by the parent of k. There are six secondary types: α_1, α_2, β_1, β_2, γ_0, δ_0. (Notice that we do not use $\gamma_0(a), \gamma_0(b)\ldots$ individually as secondary types but collapse them all into the secondary type γ_0. A similar remark holds for the δ_0 secondary type.) The secondary type of a position k will be denoted by Stype(k). For a given formula tree we shall use Γ_0 and Δ_0 to denote the sets of positions of secondary type γ_0 and δ_0 respectively. (The reader should not confuse uses of Γ and Δ as meta-variables for sets of formulae in sequents with its use here as a set of positions.)

The principal and secondary types of the positions of the formula tree of Figure 3–1 are shown below.

k	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
pol(k)	0	1	1	0	1	1	0	1	0	1	0
Ptype(k)	α	α	β	–	–	β	–	–	α	–	–
Stype(k)	–	α_1	α_1	β_1	β_2	α_2	β_1	β_2	α_2	α_1	α_2

Recall that we imposed a duplication restriction on the sequent-based proof search considered in the previous chapter. We stipulated that no duplication was to occur at any reduction other than at the reduction of generative S-formulae. We argued that this restriction does not compromise completeness. Therefore we can restrict ourselves to the representation of derivations in which only one image of a subformula, lab(k), of the endsequent can appear in any sequent unless explicitly duplicated by a reduction of a generative S-formula: lab(k'), such that the former is a subformula of the latter; *i.e.*, $k' \ll k$.

A *multiplicity* controls how many distinct instances of particular subformulae may occur in a derivation. A position k represents a subformula lab(k) as described above. We distinguish different instances of this formula by indexing the position thus: k^κ, where κ is a sequence of positive integers. We arrange

that $\mathrm{lab}(k^\kappa)$ represents a distinct instance of $\mathrm{lab}(k)$ for each distinct index κ. That is:

$$\mathrm{lab}(k^\kappa) = \mathrm{lab}(k^\tau) \quad \text{iff} \quad \kappa = \tau.$$

Multiplicities are the means by which we generate appropriate indices.

The following definitions are introduced for a given formula tree for a given signed formula $X = \langle A, 0\rangle$. A function μ from Γ_0 to the natural numbers is called a *multiplicity* for X; it serves to encode the number of instances of subformulae of X in the scope of a quantifier of universal force considered within a derivation.

If μ is a multiplicity for X we define the (indexed) formula tree for the *indexed formula* X^μ as a tree of indexed positions of the form: k^κ, where k is a position of the basic formula tree for X and κ is a sequence of positive integers defined in the manner described below. Let $k_1 \ll k_2 \ll \cdots \ll k_n \trianglelefteq k$, $1 \leq n$, be all those elements of Γ_0 that dominate k in the formula tree for X. The indexed position k^κ is a position of the indexed formula tree for X^μ provided:

1. k is a position of the formula tree for X.

2. $\mu(k_i) \neq 0$, $1 \leq i \leq n$.

3. $\kappa = m_1 m_2 \cdots m_n$ where $1 \leq m_i \leq \mu(k_i)$, $1 \leq i \leq n$.

We shall use $\kappa \prec \tau$ to denote that κ is a proper initial sequence of τ. The ordering on the underlying tree is extended to the indexed tree as follows: for indexed positions k^κ and l^τ,

$$k^\kappa \ll^\mu l^\tau \qquad \text{iff} \qquad k \ll l \quad \text{and} \quad \kappa \preceq \tau$$

i.e., k must dominate l in the unindexed formula tree, and κ must be an initial, but possibly not proper, sequence of τ. The polarity, $\mathrm{pol}(k^\kappa)$, of an indexed position k^κ is taken to be the same as the polarity of its underlying unindexed position k; *i.e.*, $\mathrm{pol}(k^\kappa) = \mathrm{pol}(k)$. The label, $\mathrm{lab}(k^\kappa)$, of an indexed position k^κ is defined inductively as follows:

1. $\mathrm{lab}(k_0) = A$, if k_0 is the root position of the formula tree.

2. If $\mathrm{lab}(k^\kappa) = B \wedge C$, and k_1^κ, k_2^κ are the children of k^κ, then $\mathrm{lab}(k_1^\kappa) = B$ and $\mathrm{lab}(k_2^\kappa) = C$.

3. If $\mathrm{lab}(k^\kappa) = B \vee C$, and k_1^κ, k_2^κ are the children of k^κ, then $\mathrm{lab}(k_1^\kappa) = B$ and $\mathrm{lab}(k_2^\kappa) = C$.

4. If $\mathrm{lab}(k^\kappa) = B \Rightarrow C$, and k_1^κ, k_2^κ are the children of k^κ, then $\mathrm{lab}(k_1^\kappa) = B$ and $\mathrm{lab}(k_2^\kappa) = C$.

5. If $\mathrm{lab}(k^\kappa) = \neg B$, and k_1^κ is the child of k^κ, then $\mathrm{lab}(k_1^\kappa) = B$.

6. If $\mathrm{lab}(k^\kappa) = \forall x B$, and k_1^τ is a child of k^κ, for some τ, $\kappa \preceq \tau$, then $\mathrm{lab}(k_1^\tau) = B[k_1^\tau/x]$.

7. If $\mathrm{lab}(k^\kappa) = \exists x B$, and k_1^τ is a child of k^κ, for some τ, $\kappa \preceq \tau$, then $\mathrm{lab}(k_1^\tau) = B[k_1^\tau/x]$.

That is, we use the position itself as a marker for where substitutions can be performed for individual variables. Positions of γ_0 and δ_0-type appear in the atoms labelling the atomic positions of the indexed formula tree in place of what otherwise would be free individual variables. One may think of the positions used in this way as a special structured set of parameters for our sequent proof theory but we prefer another interpretation given in §3.4.

The notation $\mathrm{sform}(k^\kappa)$ is extended to indexed positions in the obvious way, namely:

$$\mathrm{sform}(k^\kappa) = \langle \mathrm{lab}(k^\kappa)\,, \mathrm{pol}(k^\kappa) \rangle.$$

Consequently, since the polarity of an indexed position k^κ, and the structural form of its label, is identical to the polarity and form of the label of the underlying position k, k^κ inherits the types (both principal and secondary) of k. We use $\Gamma_0(\mu)$ and $\Delta_0(\mu)$ to denote the sets of indexed positions of (secondary) type γ_0 and δ_0 respectively, in a given (indexed) formula tree for X^μ.

We shall use u and v, possibly subscripted, as meta-variables ranging over indexed positions when we are not interested in the index, and drop the superscript on $\ll^{\mu}$. Moreover we use $\alpha, \alpha_1, \alpha_2, \beta, \ldots$, *etc*, to denote arbitrary indexed positions of that (principal or secondary) type. We shall feel free to use such notation as α^{κ} to denote an indexed position of α-type when we wish to identify the index. Henceforth we shall refer to indexed positions simply as positions.

We adopt a number of typographical conventions to ease the reader's task in following the examples given in the text. These conventions are:

- Indices (sequences of positive integers) are written as strings. For instance, the string 121 represents the three-element sequence consisting of "1" followed by "2" followed by "1." Consequently, $12 \prec 121$. We will have no need to consider multiplicities higher than 9 in the discussion or the examples presented in this thesis.

- We omit the index on a position if it is the empty sequence.

- Elements of $\Gamma_0(\mu)$ are distinguished with an overbar. These positions play a crucial rôle in the sequel.

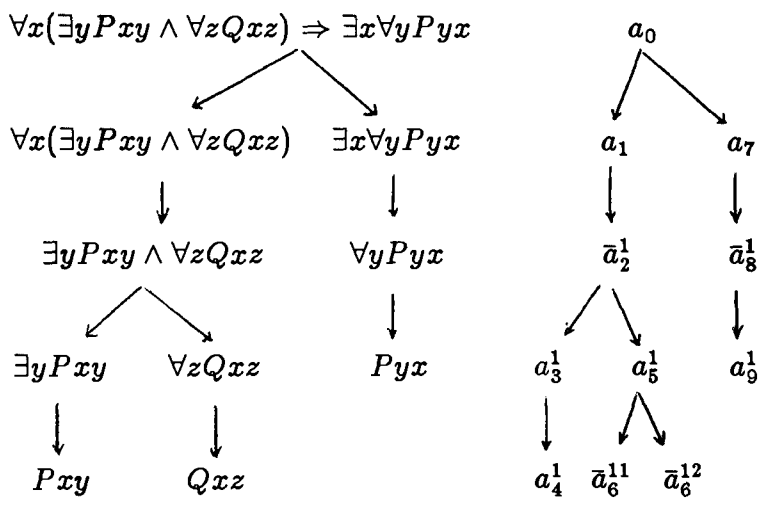
An indexed formula tree for the signed formula:

$$\langle \forall x(\exists y Pxy \wedge \forall z Qxz) \Rightarrow \exists x \forall y Pyx, 0 \rangle$$

is shown in Figure 3–2. We have taken the multiplicity to be $\mu(\bar{a}_2) = 1$, $\mu(\bar{a}_6) = 2$ and $\mu(\bar{a}_8) = 1$.

Notice how the free variables of subformulae in the formation tree are replaced by positions in the labels of the indexed formula tree. This is a consequence of the definition of the labels of indexed positions given above. We shall step through the consequences and technical motivations for these definitions with reference to the example. Our explanation will be inductive, following the definitions.

Consider a signed formula $\langle A, 0 \rangle$ representing the sequent $\longrightarrow A$. Consider a free positive subformula of $\langle A, 0 \rangle$ whose major symbol is a universal quantifier;



u	pol(u)	lab(u)	Ptype(u)	Stype(u)
a_0	0	$\forall x(\exists y Pxy \wedge \forall z Qxz) \Rightarrow \exists x \forall y Pyx$	α	–
a_1	1	$\forall x(\exists y Pxy \wedge \forall z Qxz)$	γ	α_1
$\bar{a}_2^1$	1	$\exists y P\bar{a}_2^1 y \wedge \forall z Q\bar{a}_2^1 z$	α	γ_0
a_3^1	1	$\exists y P\bar{a}_2^1 y$	δ	α_1
a_4^1	1	$P\bar{a}_2^1 a_4^1$	–	δ_0
a_5^1	1	$\forall z Q\bar{a}_2^1 z$	γ	α_2
$\bar{a}_6^{11}$	1	$Q\bar{a}_2^1 \bar{a}_6^{11}$	–	γ_0
$\bar{a}_6^{12}$	1	$Q\bar{a}_2^1 \bar{a}_6^{12}$	–	γ_0
a_7	0	$\exists x \forall y Pyx$	γ	α_2
$\bar{a}_8^1$	0	$\forall y Py\bar{a}_8^1$	δ	γ_0
a_9^1	0	$Pa_9^1\bar{a}_8^1$	–	δ_0

Figure 3–2: An indexed formula tree.

i.e., a positive subformula of the form: $\forall xB$. Let the position that represents this subformula in the (basic) formula tree for $\langle A, 0\rangle$ be l. We have:

$$\begin{aligned} \text{pol}(l) &= 1 \\ \text{lab}(l) &= \forall xB \\ \text{sform}(l) &= \langle \forall xB, 1\rangle \\ \text{Ptype}(l) &= \gamma \end{aligned}$$

(Example: $\forall xB$ is $\forall x(\exists y Pxy \wedge \forall z Qxz)$ and l is a_1 in Figure 3–2.)

Since the free subformula $\forall xB$ occurs positively in $\longrightarrow A$ (negatively in A) instances of it will occur as antecedent S-formulae. Consider such an instance $\forall xB'$ and suppose that it is represented by an indexed position l^κ in an indexed formula tree for $\langle A, 0\rangle^\mu$. This constitutes the inductive assumption for the explanation. We have:

$$\begin{aligned} \text{pol}(l^\kappa) &= 1 \\ \text{lab}(l^\kappa) &= \forall xB' \\ \text{sform}(l^\kappa) &= \langle \forall xB', 1\rangle \\ \text{Ptype}(l^\kappa) &= \gamma \end{aligned}$$

The S-formula $\langle \forall xB', 1\rangle$ represented by l^κ is generative in the sense that multiple distinct instances of B' could occur in derivations. Instances are formed by the substitution of parameters for the free occurrences of x in B'. It is the only free variable in B' by the inductive assumption. The other free variables in the free formula B will have been treated as we are now treating x already.

Let k be the child of l in the basic formula tree. We have:

$$\begin{aligned} \text{pol}(k) &= 1 \\ \text{lab}(k) &= B \\ \text{sform}(k) &= \langle B, 1\rangle \\ \text{Stype}(k) &= \gamma_0 \end{aligned}$$

(Example: k is $\bar{a}_2$ of the unindexed formula tree which is not shown in Figure 3–2.)

By choosing a multiplicity with $\mu(k) = 1$ we prescribe that at most one distinct instance of B' may be considered in the derivation. (Remember that B' contains the free variable x.) Put another way, the language represented by the formula tree indexed by that multiplicity can only describe derivations in which at most one instance of B' is considered. We generate a distinct name for this instance by the indexing method. In the indexed formula tree with this multiplicity l^κ has only one child: k^τ, where τ is the sequence $\kappa 1$. k^τ names the instance of B'. We have:

$$\begin{aligned} \mathrm{pol}(k^\tau) &= 1 \\ \mathrm{Ptype}(k) &= \gamma_0 \end{aligned}$$

(Example: k^τ is $\bar{a}_2^1$ since $\kappa = \emptyset$.) The only issue that remains open is to define the instance of B' named by $k^{\kappa j}$. We have some flexibility in this choice since any parameter or constant can be substituted for x according to the (inverted) universal rules. Instead of attempting to choose at this stage, we delay the choice and indicate the potential for a substitution by replacing the free variable by the distinct position $k^{\kappa j}$ itself. The effect of this is that the atomic formulae of B' contain the position $k^{\kappa j}$. We now have:

$$\begin{aligned} \mathrm{lab}(k^\tau) &= B'[k^\tau/x] \\ \mathrm{sform}(k^\tau) &= \langle B'[k^\tau/x], 1\rangle \end{aligned}$$

(Example: The label of $\bar{a}_2^1$ is $\exists y P\bar{a}_2^1 y \wedge \forall z Q\bar{a}_2^1 z$.)

This completes the inductive construction. We have applied the rule:

6: If $\mathrm{lab}(k^\kappa) = \forall x B$, and k_1^τ is a child of k^κ, for some τ, $\kappa \preceq \tau$, then $\mathrm{lab}(k_1^\tau) = B[k_1^\tau/x]$

of the definition of the labels of indexed positions.

The reason for the inclusion of positions in the labels of indexed positions is technical. In the sequel we shall use a mapping of elements of $\Gamma_0(\mu)$ (the positions with an overbar) into the set $\Gamma_0(\mu) \cup \Delta_0(\mu)$. This mapping indicates the coherence in choice parameter necessary to ensure that certain atomic formula

are identical. We wish to compute these mappings using a unification algorithm. Using the formulation of labels above we can simply unify the labels of the atomic positions and deduce the mapping from that.

REMARKS. The notion of multiplicity and indexed formula given here differ from those given by Bibel in [Bib80,Bib82c]. Bibel's multiplicity can be said to be a function from the positions with $\mathrm{Ptype}(k) = \gamma$ instead of those with $\mathrm{Stype}(k) = \gamma_0$ if formulated using the notions developed above. This is because he considers the quantified formula to be duplicated ($\forall x B'$) rather than the formula quantified (B'). A quantifier may only be eliminated via the introduction of one distinct parameter "associated" with it. Instead of using positions for free variables as we do, he indexes the variables themselves. In his scheme (stretching our terminology):

$$\begin{aligned} \mathrm{lab}(l^\tau) &= \forall x^\tau B'[x^\tau/x^\kappa] \\ \mathrm{Ptype}(l^\tau) &= \gamma \end{aligned}$$

(He does not formulate the notion of label as we do.) He is then able to leave the free variable in place when the quantifier is reduced and use it, as do we, to indicate the coherence necessary in the parameter substitution. If he formulated the notion of label the clause for quantifiers would look like this:

6. If $\mathrm{lab}(k^\kappa) = \forall x B$, and k_1^κ is a child of k^κ, then $\mathrm{lab}(k_1^\kappa) = B$

The other clauses would be problematic however since we would have to consider as separate cases the situations where the immediate subformula of a label is a quantified formula. Despite the technical problems his method is essentially equivalent to ours *for dealing with classical quantifiers only.*

From a sequent point of view Bibel's duplication method is the rule (read from conclusion to premise):

$$\frac{\Gamma,\ \forall x' B[x'/x],\ \forall x B \longrightarrow \Delta}{\Gamma,\ \forall x B \longrightarrow \Delta}$$

and the reduction like this:

$$\frac{\Gamma,\ C[c/y] \longrightarrow \Delta}{\Gamma,\ \forall y C \longrightarrow \Delta}$$

whilst our method formalises the rule (read from conclusion to premise):

$$\frac{\Gamma, \forall yC, C[c/y] \longrightarrow \Delta}{\Gamma, \forall yC \longrightarrow \Delta}$$

We do not consider the duplication of the formula $\forall yC$ to be a duplication at all since we can already represent it by a position. The duplication occurs when and if the formula is reduced again to introduce another instance of C. Our concern for building a representation language for sequent derivations forces us to the formulation described above.

Multiplicities are used to support the solution to problems of order dependence developed below. Our formulation gives a general account of the distinct formula that make up a sequent derivation whilst Bibel's does not. In his writings, even though he developed the Connection Calculus via a study of sequent proofs, he considers this technique basically a better alternative to Skolemisation [CL73] rather than capturing proof-theoretic properties concerning the relationship of formulae with a derivation. We argue that this latter more abstract view can have significant benefits. The pay-off for our formulation is that we are able to apply these techniques in the sequel to modal operators (Part II) where no variables are concerned at all, and intuitionistic logic (Part III). (END OF REMARKS.)

3.2.4 Summary.

We have introduced various notions based around that of a formula tree and its positions for the removal of the notational redundancy present within the sequent search space. Indexed formula trees provide a general language for the representation of the formulae that can make up a derivation of a given endsequent in terms of the structure of the endsequent. The technique relies on the fact that the cut-free sequent calculus possesses the subformula property.

From a practical point of view, positions can be interpreted as pointers to a single concrete representation of the endsequent in the database of the proof procedure. We note that all the information concerning types, polarities and

labels can be calculated before any attempt at a proof is made. When interpreted in this way we have achieved a theoretical exposition of the technique of structure sharing [BM72]: a technique developed to reduce the notational redundancy of resolution-based systems.

The essence of the technique for quantifiers is due to Bibel. We have reformulated it and claim that our formulation is more carefully motivated in terms of the structure of sequent derivations. Having separated out the technique from the others in the Connection Calculus, it becomes possible to apply it within proof procedures not based on any matrix method. Any serious implementation of a sequent or analytic tableau proof procedure should consider utilising the technique for the representation of intermediate derivations (*eg.*, [OS86]).

An interesting use of multiplicities to formulate decision procedures for propositional modal logics is considered in Chapter 7.

3.3 Paths, connections and relevance.

In this section we address the problem of relevance within the sequent search space. Recall that when extending a derivation by reducing one of its leaves we have to decide which S-formula of the leaf actually to reduce. We have no criteria for choosing one S-formula over another. There are a number of obvious strategies for ordering these OR-choices. Smullyan [Smu68] considers a number of such refinements for his analytic tableau proof methods which are notational variants of the sequent calculi we consider. In general such strategies amount to a preference for α-type and δ-type S-formulae over β-type and γ-type ones. Oppacher and Suen [OS86] have described some related strategies.

None of these strategies overcome the fundamental problem which is the emphasis on the outermost form of an S-formula and not its internal structure. Any strategy based on this sort of information cannot identify whether a given reduction will hasten the construction of a basic sequent or not. The connective-driven nature of sequent search together with the method of dealing with parameter

choices are the fundamental drawbacks of sequent-based proof procedures (see *eg.*, [BT75,D84,OS86]).

Robinson [Rob65] shifted the emphasis from connectives to the *connection* with the basic "clash" of the resolution rule of inference. His *unification* algorithm also removed the need to choose parameters at the reduction of quantifiers. Unfortunately the method he developed of utilising these fundamental ideas requires the use of a severe normal-form which can introduce a large amount of redundancy before the proof search even begins (see [And81]). Moreover, the particular normal-form does not exist for many of the logics in use today. This issue forms a serious barrier to the application of resolution to non-classical logics. At best the methods are of use within a restricted class of formulae of the non-classical logic. Typically this class includes precisely those formulae that can be put in the particular normal-form required by resolution. A good case in point is Fariñas-del-Cerro's modal resolution systems [Far86].

Some authors have tried to remove this normal-form restriction. The "non-clausal" resolution rule of Manna and Waldinger [MW80] and Murray [Mur82] is one such proposal. This method forms the basis for one of the alternative modal proof methods reviewed in Part II.

Andrews [And81] and Bibel [Bib81,Bib82a] have developed a more subtle basis for retaining the sentential connectives, and in Bibel's case the quantifiers also, whilst still gaining the benefits of connective-driven search. The methods are based on the notion of a *path* through a formula. This notion is quite common in proof-theoretic accounts of logic but was proposed for use in automated proof search explicitly by Prawitz [Pra60] in his "improved" proof procedure for formulae in conjunctive normal-form. The combination of paths with connections is the crucial blend.

At first sight the methods of Andrews and Bibel look quite adhoc. It is perhaps this, together with their complexity, that has hindered an appreciation of the importance of these techniques for automated proof search. We hope that our method of presentation will help to change this situation.

In the first section we define the notion of a path through an (indexed) formula and relate it to sequent-based notions. In the second section we introduce the idea of a connection. Finally we summarise the differences between our formulation of these notions and those of Bibel and Andrews.

3.3.1 Paths as sequents.

IMPORTANT NOTATIONAL POINT. We warn the reader that we shall systematically abuse our notation and use the names of types to denote arbitrary (un)indexed positions of that type within formal definitions such as the definition of the notion of path below. In particular, if we say: "if s, γ^κ is a path..." we mean that "if s, u is a path, and $\mathrm{Ptype}(u) = \gamma$,..." Furthermore, in this context we shall use $\gamma_0^{\kappa j}$ to denote the child of u. Similar abuses are extended to the other types. We shall include indices explicitly where necessary. (END OF POINT.)

Let X^μ be an indexed formula. A *path* through X^μ is a subset of the positions of its formula tree defined below. We shall use s and t, possibly subscripted, to denote paths, and adopt the notation s, u to denote the path (set) $s \cup \{u\}$. The set of paths through X^μ, is the smallest set such that:

1. $\{ k_0 \}$ is a path, where k_0 is the root position of the formula tree for X^μ;
2. if s, α^κ is a path, so is $(s \setminus \{\alpha^\kappa\}), {\alpha_1}^\kappa, {\alpha_2}^\kappa$;
3. if s, β^κ is a path, so are $(s \setminus \{\beta^\kappa\}), {\beta_1}^\kappa$ and $(s \setminus \{\beta^\kappa\}), {\beta_2}^\kappa$;
4. if s, γ^κ is a path, so is $s, {\gamma_0}^{\kappa j}$, for any j, $1 \leq j \leq \mu(\gamma_0)$;
5. if s, δ^κ is a path, so is $(s \setminus \{\delta^\kappa\}), {\delta_0}^\kappa$.

The path:

$$(s \setminus \{\alpha^\kappa\}), {\alpha_1}^\kappa, {\alpha_2}^\kappa$$

is said to have been *obtained by reduction on* α^κ from s, α^κ. Similarly in the other cases. Notice that in the generative case: γ, there is a choice as to which

child of the position to introduce. The children differ solely in the last element of their indices.

The definition can be appreciated immediately if one remembers that each position represents a signed formula, and that a signed formula represents either an antecedent or succedent S-formula depending on its polarity. A set of signed formulae therefore represents a sequent. Formally, a set of positions s represents the sequent:

$$\Gamma_s \longrightarrow \Delta_s$$

where:

$$\Gamma_s \stackrel{\text{df}}{=} \{\, \text{lab}(u) \mid u \in s,\ \text{pol}(u) = 1 \,\}$$
$$\Delta_s \stackrel{\text{df}}{=} \{\, \text{lab}(u) \mid u \in s,\ \text{pol}(u) = 0 \,\}$$

We can interpret the α clause of the definition of path as referring to sequents. Consider a path s, u for some α-type position u. If $\text{pol}(u) = 1$ then $\text{sform}(u)$ represents an antecedent formula: $\text{lab}(u)$. Moreover, $\text{sform}(u)$ is a signed formula of α-type. Hence $\text{lab}(u)$ is a formula either of the form: $A \wedge B$, or of the form $\neg A$ (see §3.1.1, Table 3–1.) The path therefore represents one of the two sequents:

$$\Gamma_s,\ A \wedge B \longrightarrow \Delta_s \qquad \text{or} \qquad \Gamma_s,\ \neg A \longrightarrow \Delta_s$$

Suppose u_1 and u_2 are the two children of u. We have:

1. If $\text{lab}(u) = A \wedge B$, then $\text{lab}(u_1) = A$ and $\text{lab}(u_2) = B$.

2. If $\text{lab}(u) = \neg A$, then $\text{lab}(u_1) = \text{lab}(u_2) = A$.

The α clause says that if s, u is a path, so is $(s \setminus \{u\}), u_1, u_2$. Consequently, the reduced path is one of:

$$\Gamma'_s,\ A,\ B \longrightarrow \Delta'_s \qquad \text{or} \qquad \Gamma'_s \longrightarrow A,\ \Delta'_s$$

where Γ'_s is $\Gamma_s \setminus \{u\}$ and Δ'_s is $\Delta_s \setminus \{u\}$. The reduced path represents precisely the sequent that would result from an inverted application of one of the rules:

$$\frac{\Gamma,\ A,\ B \longrightarrow \Delta}{\Gamma,\ A \wedge B \longrightarrow \Delta}\ \wedge \longrightarrow \qquad \frac{\Gamma \longrightarrow A,\ \Delta}{\Gamma,\ \neg A \longrightarrow \Delta}\ \neg \longrightarrow$$

on the sequent represented by the original path, provided that no duplication is allowed. Furthermore, the root path: $\{ k_0 \}$, where k_0 is the root position of $X = \langle C, 0 \rangle$ represents the sequent:

$$\longrightarrow C$$

In fact, the set of paths through X are simply (representations of) the set of sequents constructable from this endsequent, with a restricted possibility for duplication at the reduction of antecedent S-formulae of the form: $\forall x A$, and succedent S-formulae of the form: $\exists x A$ (*i.e.*, the γ-type positions).

There are two things to note about the structure of these sequents and their generation as paths. Firstly, they contain positions in place of free variables or parameters. Secondly, there is no proviso on the application of δ rules. Notice that these differences appear only for first-order formulae not propositional ones. We showed in Chapter 1 how the atomic paths can be visualised in terms of a nested two-dimensional matrix (at least for propositional formulae).

We consider the propositional case first. An *atomic* path through X (there is no multiplicity since X is propositional) is a path containing only atomic positions. There are a finite number of atomic paths through X since the formula tree for X is finite and each reduction *replaces* a position by its children to form the reduced path. The atomic paths are the set of leaves of what we shall call a *complete derivation*. This is a derivation that has been extended as far as possible by reducing all non-atomic S-formulae in the leaves.

3.3.2 Connections.

A *connection* is simply a pair of atomic positions in some path through X^μ whose labels have the same predicate symbol but different polarities. (In propositional logic the labels are therefore identical.) Since the positions have different polarities one is a positive occurrence and the other a negative occurrence of some proposition.

A set of connections is said to *span* X^μ just in case every atomic path through X^μ contains a connection from the set. Andrews [And81] and Bibel [Bib81] prove that if $X = \langle C, 0 \rangle$ where C is a propositional formula, then the existence of a spanning set of connections in X ensures the validity of C (and vice versa). We can now see why. If every atomic path s contains a connection it represents a sequent of the form:

$$\Gamma_s,\ A \longrightarrow A,\ \Delta_s$$

for some proposition A. Consequently any complete derivation of $\longrightarrow C$ is a proof.

In the case of propositional logic at least, this is a characterisation of the existence of a sequent proof of a formula (and hence its validity) in terms of the existence of a spanning set of connections in the formula. No mention is made of actually constructing the proof. Put another way, the sequent calculus can be seen as a method of checking that every atomic path through the formula contains a connection. It is a very inefficient method of checking these paths. Better methods are developed by Bibel [Bib82a,Bib77,HB82] and Andrews [And81]. The common feature of these path-checking methods that makes them improvements on the sequent method is their emphasis on connections as opposed to paths. The sequent method (and natural deduction methods in general) enumerate paths and then check to see if they contain connections. Matrix methods identify a connection first, then eliminate from consideration all the paths that contain that connection. The paths are said to have been "checked." The consideration of paths is driven by the identification of connections and not vice versa. In sequent terms, no reduction is performed that does not directly lead to the introduction of the two atomic S-formulae corresponding to a connection. The problem of relevance is removed from the matrix search space.

EXAMPLE. We redo the propositional example given in Chapter 1 in order to illustate the relationship between path-checking and the theoretical ideas developed above. We consider the formula:

$$(P \Rightarrow Q) \wedge (Q \Rightarrow R) \Rightarrow (P \Rightarrow R)$$

A formula tree for the signed formula:

$$\langle (P \Rightarrow Q) \wedge (Q \Rightarrow R) \Rightarrow (P \Rightarrow R), 0 \rangle$$

was developed in §3.2. The basic formula tree is shown in Figure 3–1 and the assignment of polarities, labels and types developed throughout that section. In the introduction we reduced the formula to its negation normal form:

$$(P \wedge \neg Q) \vee (Q \wedge \neg R) \vee (\neg P \vee R)$$

in order to identify the atomic paths through using a matrix as a visual aid. We now have more sophisticated methods of stating the matrix representation of a formula. The matrix representation of a signed formula X is defined inductively as follows:

- If X is of α-type (*eg.*, $\langle B \Rightarrow C, 0 \rangle$), the matrix representation of X is a 1×2 matrix with the two components of X (*eg.*, $\langle B, 1 \rangle$ and $\langle C, 0 \rangle$) as the two columns.

- If X is of β-type (*eg.*, $\langle B \Rightarrow C, 1 \rangle$), the matrix representation of X is a 2×1 matrix with the two components of X (*eg.*, $\langle B, 0 \rangle$ and $\langle C, 1 \rangle$) as the two rows.

- Otherwise, if X is non-atomic, the matrix representation of X is a 1×1 matrix with the single component of the formula the submatrix. If X is atomic, it is said to be its own matrix representation.

The reader can check using the types of the positions given for the formula tree for our signed example that its matrix representation (via this definition) is:

$$\begin{pmatrix} P^0 \\ Q^1 \end{pmatrix} \quad \begin{pmatrix} Q^0 \\ R^1 \end{pmatrix} \quad (P^1 \qquad R^0)$$

or in terms of positions:

$$\begin{pmatrix} a_3 \\ a_4 \end{pmatrix} \quad \begin{pmatrix} a_6 \\ a_7 \end{pmatrix} \quad (a_9 \qquad a_{10})$$

The atomic paths through the matrix are as given in the introduction. We leave it to the reader to run the inductive definition of paths given above on the formula tree of Figure 3–1 to verify this. The atomic paths are:

$$\{ P^0, Q^0, P^1, \mathcal{R}^0 \}$$
$$\{ P^0, R^1, P^1, \mathcal{R}^0 \}$$
$$\{ Q^1, Q^0, P^1, \mathcal{R}^0 \}$$
$$\{ Q^1, R^1, P^1, \mathcal{R}^0 \}$$

or, more precisely in terms of positions:

$$\{ a_3, a_6, a_9, a_{10} \}$$
$$\{ a_4, a_6, a_9, a_{10} \}$$
$$\{ a_3, a_7, a_9, a_{10} \}$$
$$\{ a_4, a_7, a_9, a_{10} \}$$

The three connections identified in the introduction were:

$$\{ P^0, P^1 \}$$
$$\{ Q^1, Q^0 \}$$
$$\{ R^1, R^0 \}$$

or in terms of positions:

$$\{ a_3, a_9 \}$$
$$\{ a_4, a_6 \}$$
$$\{ a_7, a_{10} \}$$

Notice that the set spans the formula. We conclude that it is propositionally valid. The reader might like to check that these three connections form the distinguished atomic formulae in any sequent proof of the formula by performing the proof in terms of positions instead of formulae. Start with the root path $\{a_0\}$, look at its type, and reduce it according to the definition of path given (or alternatively translate everything into sequents via the signed formulae sform(k)).

In Chapter 1 we outlined how directed, connection-driven search procedures can be defined. (END OF EXAMPLE.)

We have considered the case of propositional logic. We consider the first-order case below. First we review the differences between our formulation of the notion of path and that of Andrews and Bibel.

3.3.3 Summary.

We have introduced the notions of "path" and "connection" and outlined how they are sufficient to overcome the problems of relevance within the sequent search space. We have shown how the sequent calculus can be seen as an inefficient method of enumerating paths through a formula and subsequently checking to see if the paths constructed are closed. Replacing that type of search by a directed search for connections ensures that only those reductions necessary for the realisation of connections are enumerated. Connections can be used to "look inside" the structure of S-formulae and calculate which reductions to perform (if any) to close a leaf.

Our formulation of path is somewhat different from that of Andrews and Bibel. Their definitions only capture what we have called an atomic path, partly because of the restriction to negation normal-form in the theoretical discussion. Our formulation makes the relationship with sequent-based ideas explicit.

In Parts II we adapt the definition of path to suit the extended modal languages. The arguments of this section are formalised completely within the justification of the modal matrix characterisations. The correctness and completeness proofs for these extended matrix systems (Chapter 6) are directly based on such proofs for sequent and tableau systems. The generalisation of the notion of path to relate it to these standard ideas permits quite straightforward meta-theoretic arguments about the matrix systems themselves. This is crucial if the matrix methods are to be extended successfully to other logics.

3.4 Reduction orderings and order dependence.

So far we have removed the notational redundancies and those of relevance from the sequent search space by means of formula trees, paths and connections. The basic matrix framework is in place. In the last section we outlined how these notions alone are sufficient to characterise validity in classical propositional logic. We come now to the last problem identified within the sequent systems: the order dependence induced in the search space by the quantifier rules.

For propositional logic we have no need to consider the internal structure of derivations at all. The existence of a spanning set of connections for A means that any complete derivation must be closed (*i.e.*, be a proof of $\longrightarrow A$) and therefore entails the classical validity of A. For first-order logic however the internal structure of derivations is important since there are constraints imposed on the application of certain quantifier rules.

First we adapt the notion of a complete derivation to first-order logic where duplication of quantified formula (γ_0 and δ_0-type formulae/positions) is permitted. We can no longer make sense of the phrase "reduce a sequent/path to atomic formulae," since there is the possibility of duplicating subformulae arbitrarily. The notion of multiplicity can be used to control this duplication. For a given multiplicity it does make sense to define a complete derivation and hence the notion of atomic path. Put simply: a path is atomic if every duplication sanctioned by the multiplicity has been performed and the results reduced to atomic positions. We formalise this notion fully in Part II, Chapter 6, when we deal with modal logics.

Consider the signed formula:

$$\langle \forall x(\exists y Pxy \wedge \forall z Qxz) \Rightarrow \exists x \forall y Pyx, 0 \rangle,$$

An indexed formula tree for this signed formula was given in §3.2.3, Figure 3–2 above. The matrix of this formula contains only one path which, with the

multiplicity given, contains four atomic elements as follows:

$$\left\{ a_4^1, \bar{a}_6^{11}, \bar{a}_6^{12}, a_9^1 \right\};$$

or in terms of labels:

$$\left\{ P\bar{a}_2^1 a_4^1, Q\bar{a}_2^1 \bar{a}_6^{11}, Q\bar{a}_2^1 \bar{a}_6^{12}, Pa_9^1 \bar{a}_8^1 \right\};$$

or in terms of atomic formulae:

$$\{ Pxy, Qxz, Qxz', Py'x' \},$$

where we have "primed" variables to indicate that they are distinct. The richness of the second representation given in terms of labels compared with the poverty of this last representation provides some justification for the complex definition of labels used.

There is only one path since there are no β-type subformulae. Recall that positions distinguished with an overbar correspond to γ_0-type subformulae: the immediate subformulae of generative subformulae. This path contains a connection: $\{a_4^1, a_9^1\}$, since the labels for these two positions have the same predicate symbol (P) and different polarities. Can we conclude that the formula is valid?

Consider the following sequent "pseudo-derivation," where instead of parameters we use positions of the formula tree instead (*i.e.*, labels instead of bonafide subformulae of the endsequent):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P\bar{a}_2^1 a_4^1,\ Q\bar{a}_2^1 \bar{a}_6^{11},\ Q\bar{a}_2^1 \bar{a}_6^{12} \longrightarrow Pa_9^1 \bar{a}_8^1}{P\bar{a}_2^1 a_4^1,\ Q\bar{a}_2^1 \bar{a}_6^{11},\ Q\bar{a}_2^1 \bar{a}_6^{12} \longrightarrow \forall y P y \bar{a}_8^1}}{P\bar{a}_2^1 a_4^1,\ Q\bar{a}_2^1 \bar{a}_6^{11},\ Q\bar{a}_2^1 \bar{a}_6^{12} \longrightarrow \exists x \forall y P y x}}{P\bar{a}_2^1 a_4^1,\ Q\bar{a}_2^1 \bar{a}_6^{11},\ \forall z Q \bar{a}_2^1 z \longrightarrow \exists x \forall y P y x}}{P\bar{a}_2^1 a_4^1,\ \forall z Q \bar{a}_2^1 z \longrightarrow \exists x \forall y P y x}}{\exists y P \bar{a}_2^1 y,\ \forall z Q \bar{a}_2^1 z \longrightarrow \exists x \forall y P y x}}{\exists y P \bar{a}_2^1 y \wedge \forall z Q \bar{a}_2^1 z \longrightarrow \exists x \forall y P y x}}{\forall x (\exists y P x y \wedge \forall z Q x z) \longrightarrow \exists x \forall y P y x}}{\longrightarrow \forall x (\exists y P x y \wedge \forall z Q x z) \Rightarrow \exists x \forall y P y x}$$

Notice that the multiplicity $\mu(\bar{a}_6) = 2$ sanctions at most two distinct instances of $Q\bar{a}_2^1 z$, the (free) immediate subformula of the generative S-formula: $\forall z Q \bar{a}_2^1 z$. This potential is represented by the two labels: $Q\bar{a}_2^1 \bar{a}_6^{11}$ and $Q\bar{a}_2^1 \bar{a}_6^{12}$. The other

generative subformulae are restricted by the multiplicity to the production of one instance of their immediate subformula only.

To transform this pseudo-derivation into a proper derivation we need to choose parameters for the positions that appear in the formulae. For the result to be a proof we must ensure that the parameters are chosen so that the antecedent S-formula: $P\bar{a}_2^1 a_4^1$, is identical with the succedent S-formula: $Pa_9^1\bar{a}_8^1$. The leaf of the transformed derivation will then be closed. We can represent this choice by a mapping ι from $\Gamma_0(\mu) \cup \Delta_0(\mu)$ into a suitable set of parameters. One immediate constraint on this mapping is that δ_0-type positions must correspond to distinct parameters; *i.e.*, for all $u, u' \in \Delta_0$,

$$\iota(u) \neq \iota(u') .$$

This is because at the reduction of a δ-type S-formula (strictly: signed formula) we are forced to introduce completely new parameters. A second constraint is that the choice of parameter at the reduction of γ-type S-formulae must not interfere with the reduction of the necessary δ-type S-formulae. We formalise these notions below.

Given a connection such as $\{a_4^1, a_9^1\}$, we consider a mapping $\sigma: \Gamma_0 \mapsto (\Gamma_0 \cup \Delta_0)$ under which the labels of the atomic positions are identical. The connection is said to be *complementary* under σ, or σ-complementary. In our example an appropriate mapping is:

$$\begin{aligned} \bar{a}_2^1 &\mapsto a_9^1 \\ \bar{a}_8^1 &\mapsto a_4^1 \end{aligned}$$

Such mappings are calculated by a unification algorithm operating on the labels of the atomic positions. The mapping σ should be interpreted as prescribing the coherence that must exist in the choice of parameters for the derivation to close. In terms of the mapping introduced from positions to parameters this is formalised as a constraint on ι: if $\sigma(u) = \sigma(u')$, then $\iota(u) = \iota(u')$. Our example mapping represents the need for the parameter chosen at the introduction of $\bar{a}_2^1$ (by the reduction of its parent) to be the same as the one chosen at the introduction of a_9^1. A similar constraint holds between $\bar{a}_8^1$ and a_4^1.

σ induces a binary relation: $\sqsubset$ on $\Delta_0(\mu) \times \Gamma_0(\mu)$, and a binary relation: $\sim$ on $\Gamma_0(\mu) \times \Gamma_0(\mu)$, with the latter constrained to be an equivalence relation.

1. If $\sigma(u) = v$ and $v \in \Gamma_0(\mu)$, then $u \sim v$.

2. If $\sigma(u) = v$ and $v \in \Delta_0(\mu)$, then $v \sqsubset u$.

3. If $v \sqsubset u$ and $u \sim u'$, then $v \sqsubset u'$.

Finally, we define the *reduction ordering* $\lhd$ as the transitive closure of the union of the formula tree ordering $\ll$ and $\sqsubset$. That is to say:

$$\lhd \stackrel{\text{df}}{=} (\ll \cup \sqsubset)^+.$$

The relation $\sqsubset$ should be interpreted as representing order constraints on the introduction of parameters. The parameter chosen at the reduction of a δ-type formula must be new. If to close a leaf of a derivation the same parameter must be used at the reduction of a γ-type formula, the former reduction had better occur before the latter else the proviso on the δ (existential) rules will prohibit the required coherence. This is reflected as a constraint on the order of introduction of the immediate subformulae of γ and δ-type formulae, *i.e.*, (the labels of) γ_0 and δ_0 positions.

In matrix-based search we do not need to choose particular parameters, nor choose a particular order in which to apply sequent rules. We are simply interested in inferring the necessary coherence any such choice of parameters must possess, and then ensuring that there is *at least one* order of reduction that respects the provisos on the quantifier rules. The condition necessary for this latter situation to pertain is that the reduction ordering induced by σ be *irreflexive*. If this is the case, σ is said to be *admissible*.

The reduction ordering induced by our mapping is shown in Figure 3–3 as a directed graph. Notice that it is cyclic. This entails that $\lhd$ is reflexive and hence that the coherence required to identify the atomic S-formula forming the connection in the leaf of the pseudo-derivation cannot be realised (at this multiplicity). In actual fact the formula is not valid.

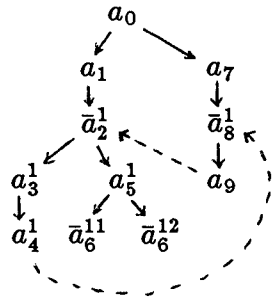


Figure 3–3: A reduction ordering as a directed graph.

EXAMPLE. We consider the example that motivated our discussion of the order problem in Chapter 2, namely:

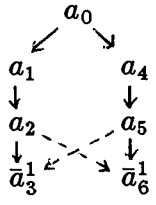
$$\langle \exists x \forall y P x y \Rightarrow \forall x \exists y P y x, 0 \rangle$$

We choose a constant multiplicity equal to 1. An indexed formula tree is shown in Figure 3–4.

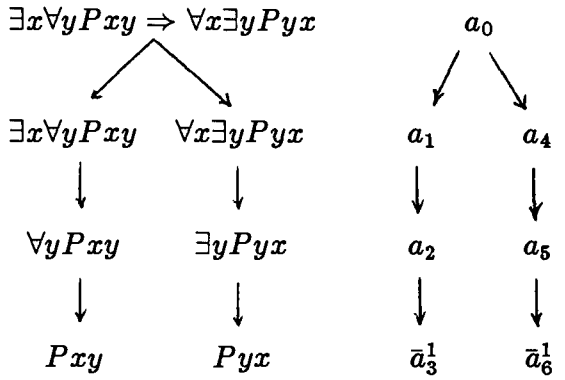
The matrix for this signed formula is simple since there are only two atomic subformulae and no β-type subformula. There is only one atomic path through the indexed formula: $\{\bar{a}_3^1, \bar{a}_6^1\}$ and these two atomic positions form a connection. The mapping σ that identifies their labels is:

$$\begin{aligned} \sigma\left(\bar{a}_3^1\right) &= a_5 \\ \sigma\left(\bar{a}_6^1\right) &= a_2 \end{aligned}$$

Consequently, $a_5 \sqsubset \bar{a}_3^1$ and $a_2 \sqsubset \bar{a}_6^1$ are induced. The reduction ordering is:



It is acyclic and hence there are sequent derivations with the right coherence in the choice of parameters to realise the connection as (part of) a basic sequent. In this case this is enough to determine the validity of the formula.



u	pol(u)	lab(u)	Ptype(u)	Stype(u)
a_0	0	$\exists x \forall y P x y \Rightarrow \forall x \exists y P y x$	α	–
a_1	1	$\exists x \forall y P x y$	δ	α_1
a_2	1	$\forall y P a_2 y$	γ	δ_0
$\bar{a}_3^1$	1	$P a_2 \bar{a}_3^1$	–	γ_0
a_4	0	$\forall x \exists y P y x$	δ	α_2
a_5	0	$\exists y P y a_5$	γ	δ_0
$\bar{a}_6^1$	0	$P \bar{a}_6^1 a_5$	–	γ_0

Figure 3–4: An indexed formula tree $\langle \exists x \forall y P x y \Rightarrow \forall x \exists y P y x, 0 \rangle$.

The permissible orders for reducing S-formulae can be read from the reduction ordering graph. The S-formula corresponding to a position in the graph must not be introduced (by the reduction of its parent) until all the other positions in the graph from which it is reachable have themselves been introduced.

In Chapter 1 we deduced these constraints by a consideration of sequent derivations directly. Here we see their representation within a matrix framework. Proof search is driven in the same way as in the propositional case: the identification of connections and the subsequent elimination of the atomic paths containing the connection. The only addition is that we must check that all the connections used so far to eliminate paths are σ-complementary under some admissible mapping σ. There is no need to consider the order in which paths (the matrix equivalent of sequents) are reduced, we simply check the atomic ones for connections and let unification do the work. (END OF EXAMPLE.)

To summarise: we have motivated Bibel's matrix characterisation of validity in first-order classical logic. The characterisation has the form: a formula A is valid if and only if there is some multiplicity μ, some admissible mapping σ and some spanning set of σ-complementary connections in $\langle A, 0\rangle^{\mu}$. Unification is used to ensure the *existence* of a correct order of sequent rule applications to produce a proof of the formula. No single concrete order need be preferred. This technique removes the order dependence, induced in the sequent search space by the quantifier rules, from the matrix search space.

3.5 Summary and discussion.

In the previous sections we have introduced the basic notions used by Bibel and Andrews in the formulation of their matrix methods for classical logic. The presentation has not been completely formal, since we do just that in the more complicated case of modal logic in Part II. We have shown how these techniques can be used to overcome the redundancies in the sequent search space analysed in Chapter 2.

Whilst the basic nature of the characterisation is due to Bibel, we believe our particular *technical* formulation of the individual techniques to be quite significant. A summary of the main points of this argument is as follows:

- The use of polarity and uniform notation.

  Bibel and Andrews restrict their theoretical discussion to formulae in negation normal-form. Whilst we recognise that this restriction is not necessary, it has

  (a) served to confuse some readers (*eg.*, Murray [Mur82] believed it to be a restriction on the scope of the matrix methods); and

  (b) their subsequent formulation of certain notions actually relies on it thus restricting the techniques theoretically to logics which admit such a normal form.

  An example of the latter is the notion of atomic path. We have eliminated this restriction in the theory by means of the notion of *polarity.* The cost was potentially high, in terms of the complexity of the resulting formulation, since we now need to consider the entire first-order language. We overcame this problem by means of Smullyan's uniform notation [Smu68].

- Indexed formula trees.

  We have altered Bibel's notion of multiplicity in a simple way that, we believe, is more in line with the motivation: providing a uniform language for the representation of derivations which supports structure sharing implementations. In our scheme the formulae duplicated are the immediate subformulae of generative subformulae. We prefer this formulation because it allows a uniform treatment of modal operators and the properties of intuitionistic sentential connectives in the sequel.

  Andrews, Miller and Pfenning's [Mil84,Pfe84] notion of an "expansion tree" is a somewhat less syntactic version of Bibel's formula trees.

They are primarily interested in higher-order logic rather than first-order logics. They demonstrate the utility of expansion trees for the representation of derivations by showing how they encode derivations in both analytic and non-analytic proof systems. They are not primarily concerned with the efficiency of the proof systems represented in the sense that we are in this thesis. Expansion trees may turn out to be a more appropriate theoretical tool for assessing the relative efficiency of various proof systems for automated proof search. That has yet to be shown. This is an interesting topic for future research.

- Paths.

  We have generalised the notion of path so that its relationship with sequent-based notions is immediately apparent. Both Bibel and Andrews formalise what we call "atomic paths," and their formulation relies on the use of negation normal-form. In Part II we shall encounter logics where the notion of an atomic path is quite complex. This can be explained in terms of sequent calculi by noting that in those logics the reduction of certain S-formulae is strongly coupled to the reduction of other S-formulae; *i.e.*, multiple reductions occur during the application of a single sequent rule. Moreover, there are irreducible non-atomic paths. With the extra flexibility gained from our formulation of path we are able to define a notion of an atomic path that is appropriate for those logics (though perhaps we should call it a "basic path" since it contains non-atomic formulae, and to associate it more directly with "basic sequents.") The logics referred to here are the K-logics.

- Admissibility.

  Given the changes mentioned above our formulation of the way in which unification can be used to remove the order dependence in the sequent search space was bound to differ from Bibel and Andrews'

account. We note that, until recently, Andrews and his co-workers retained skolemisation in their matrix framework (see *eg.*, [And81]). Their later works ([Mil84,Pfe84]) do incorporate Bibel's technique of a reduction ordering. Both Bibel and Andrews see the use of the technique as a method of dealing specifically with the parameter conditions on quantifiers of existential force. We, on the other hand, have a more general proof-theoretic view in terms of the order dependence of S-formula reductions within sequent derivations. It is this more general view that has enabled us successfully to apply a version of the technique to remove the order dependence of more complicated sequent rules such as the modal rules and rules for negation and implication in intuitionistic logic. The technique is used in the propositional fragments of the logics as well as the quantified systems.

Many of these technical improvements were developed during the adaptation of Bibel's techniques to modal and intu itionistic logics and in an attempt to formulate the relationship of the techniques with sequent-based ideas. Some of the improvements are merely technical, some are quite fundamental as summarised above. We believe this work to be significant for the application of the ideas to improve automated proof search in logics other than those considered in the remainder of this thesis.

Part II

Automated Proof Search in Modal Logics.

Summary.

The main result presented in the thesis so far is the decomposition of Bibel's Connection Calculus [Bib80,Bib82c]: a matrix characterisation of validity for classical logic, into a set of individual techniques for overcoming problems of redundancy in sequent-based proof search. The redundancies were identified by an analysis of the search space induced by a cut-free sequent calculus. We strove to abstract the formulation of the techniques from any dependence on specific details of classical logic. Having identified and abstracted these techniques, we may now apply them *individually* as and when they are applicable. In this part of the thesis we make use of this flexibility to formulate matrix characterisations of validity for a wide class of modal logics.

Modal logics are used extensively in various branches of Artificial Intelligence and Computing Science as logics of knowledge and belief (*eg.*, [Moo80,Kon84, HM85]), logics of programs (*eg.*, [Pne77,Har79]), and for such tasks as the specification of distributed and concurrent systems (*eg.*, [HM84,Sti85b]). In many — if not all — of these applications the need arises for proof systems which facilitate efficient automated proof search. Our purpose in this part of the thesis is to develop matrix characterisations of validity for both propositional and first-order versions of the modal logics K, K4, D, D4, T, S4 and S5. Our methods extend to the varying, cumulative and constant domain variants of the quantified logics.

By judicious use of the techniques identified in Part I of this thesis, we manage to retain the basic structure of the matrix characterisation for classical logic. Proof search is reduced to a process of path-checking and complementarity tests for pairs of atomic formulae (connections). As a consequence, the matrix characterisations that we formulate here render search methods developed for use with the classical matrix characterisation, *eg.*, [Bib77,Bib82b,HB82], applicable *without change* to the modal logics. We have thus succeeded in extending perhaps *the* most efficient proof search methods developed for classical logic (see [Bib82b]) to this important class of non-classical logics. The effective automation

of these logics has been a goal of the Automated Theorem Proving community for some time.

Our method for deriving the matrix characterisations of validity for a modal logic follows the pattern established in Part I. We analyse the redundancies within the search space induced by standard cut-free sequent calculus for that logic and then adapt and apply the various techniques discussed in Part I to remove them.

This part of the thesis comprises four chapters. In the first chapter we provide a basic introduction to the syntax and semantics of the modal logics under consideration. This chapter is included for completeness and reviews well-known material. We present cut-free sequent calculi for the logics K, K4, D, D4, T, S4, and briefly discuss why similar cut-free proof systems cannot be formulated for S5 and the constant-domain variants of the first order logics. Readers familiar with such proof systems for modal logics can safely skip this chapter provided they are familiar with Fitting's generalisation of Smullyan's uniform notation for modal logic [Fit83]. This notation will be used extensively in the sequel.

In Chapter 5 we investigate the search spaces of the modal sequent systems. We conclude that inference rules for modalities introduce considerable complications over and above the problems associated with the basic sequent framework itself. We analyse the nature of these complications.

From this analysis, and using the insight gained from our discussion of classical logic in Part I, in Chapter 6 we derive matrix characterisations of validity for the modal logics under consideration. This constitutes the major result of this part of the thesis. We prove the correctness and completeness of these characterisations in that chapter.

In Chapter 7 we outline efficient proof systems based on the matrix characterisations developed showing how the problems associated with sequent-based proof search are avoided. Attention is paid to the nature of the complementarity tests which require algorithms for unification under simple equational theories.

We also outline how efficient decision procedures for the propositional fragments of the modal logics can be developed based on the matrix characterisations.

A number of authors have attempted to develop computationally efficient proof systems for the modal logics considered here, for example: [Far86,AM86a, Kon86]. We conclude this part of the thesis with a review of the main proposals in the literature. The analysis demonstrates the advantages of the matrix systems as a basis for automated proof search in modal logics.

Chapter 4

The semantics and proof theory of modal logics.

4.1 Introduction.

In this chapter we:

- provide a basic introduction to the syntax and standard (Kripke) semantics of the modal logics K, K4, D, D4, T, S4 and S5;
- extend to the modal language the uniform notation utilised in our arguments above for classical logic, and
- present cut-free sequent calculi for first-order versions of the logics K, K4, D, D4, T, S4.

We also discuss why similar (cut-free) sequent systems for S5 and the constant-domain variants of the first-order logics cannot be formulated. The material of this chapter is based on the presentation of analytic tableau systems for modal logics by Fitting in his book [Fit83]. It is included so as to make the thesis self-contained. Readers familiar with analytic proof systems for modal logics can safely skip these sections provided they are also familiar with the uniform notation defined in §4.2.3. This notation will be used extensively in the sequel.

The chapter is structured as follows. First we introduce the language and semantics of modal logic and present Smullyan and Fitting's uniform notation to simplify the metatheory (§4.2). In §4.3 we develop cut-free sequent calculi for the cumulative domain variants of the modal logics K, K4, D, D4, T, S4. We develop a calculus for a single logic first, namely S4, then infer the variants for the other logics from this basis. We prove the correctness of the S4-system explicitly to familiarise the reader with the style of proofs that arise in Chapter 6. Finally, we discuss the problems that arise in attempting to formulate similar (cut-free) sequent systems for S5 and the constant-domain variants of the first-order logics (§4.4). We conclude with a summary.

The discussion of modality is brief. We are concerned with proof-theoretic properties of calculi for these logics rather than arguing the case for the use of a particular modal logic for a particular application. The references cited above should be consulted for such arguments. The reader is referred to Hughes and Cresswell's book [HC68] for a more comprehensive discussion of modal logic itself and a bibliography.

4.2 Syntax, semantics and notation.

In this section we present the syntax and (Kripke) semantics of the modal logics under consideration. We also extend the uniform notation utilised in the previous chapter to the modal language. This extension is due to Fitting [Fit83].

4.2.1 Syntax.

The first-order language common to the modal logics considered comprises:

1. A denumerable list of n-ary predicate symbols $P^n, Q^n, \ldots$, for each natural number n.

2. A denumerable list of individual variables x, y, z, (possibly subscripted).

3. Infinitely many constants c, d, (possibly subscripted).

4. The sentential connectives $\wedge$, $\vee$, $\Rightarrow$ and $\neg$.

5. The quantifiers $\forall$ and $\exists$.

6. The modal operators $\Box$ and $\Diamond$.

As usual we assume the sets of symbols to be disjoint. We omit the arity of predicate symbols when it is clear from the context or irrelevant.

REMARKS. The above definition defines a class of languages dependent on particular choices of predicate, variable, and constant symbols. In the sequel we assume some fixed set of predicate and variable symbols and allow the set of constants of the language to vary. We use this flexibility to formalise the interpretation of our formal language in the language of models so as to define an appropriate notion of validity. If D is a set of constants, by "a modal language over D" we mean a language defined as above whose constant symbols are amongst D.

Notice that the languages we consider contain no function symbols. We make this restriction here and in the sequel for technical simplicity. Since the matrix characterisations for modal logic derived in Chapter 6 stem from considerations of the sequent calculi developed in this chapter, they characterise modal validity in languages with no function symbols also. We stress: the restriction is made for technical convenience only. In Chapter 7 we show how the restriction can be lifted. (The reader may recall that we made a similar simplifying restriction in Part I when considering classical logic.) (END OF REMARKS.)

The sentential connectives and quantifiers are given their usual interpretations. There are many common informal interpretations for the modal operators. To reflect the historical origins of the symbols we shall refer to $\Box$ and $\Diamond$ as the operators of logical *necessity* and *possibility* respectively. (Some authors use the symbols L and M for these operators respectively.)

The formation rules for the set of modal formulae are simply the formation rules for classical first-order formulae augmented by the following rule for the unary modal operators:

- If A is a formula, then so are $\Box A$ and $\Diamond A$.

We shall use A, B, C as metavariables for modal formulae. The notions of subformula and immediate subformula are extended to the modal language in the obvious way: A is the *immediate subformula* of $\Box A$ and $\Diamond A$, and A together with its subformulae are the subformulae of $\Diamond A$ and $\Box A$. The notions of *substitution*, *free* and *bound* occurrences of individual variables in formulae, and *sentences* are defined as usual.

4.2.2 Semantics.

We include for completeness the standard Kripke semantics [Kri63] for the modal logics under consideration.

A pair $\langle G, R \rangle$, comprising a non-empty set G and a binary relation R on G is called a *frame*. If we restrict R to satisfy the conditions outlined in Table 4–1 we say that $\langle G, R \rangle$ is an $\mathcal{L}$-frame, where $\mathcal{L}$ is the logic associated with that condition. The condition of *idealisation* referred to in the table is the following:

for each $w \in G$ there is some $v \in G$ such that $w \; R \; v$ holds.

Notice that the reflexive relations satisfy the idealisation condition since for all $w \in G$, $w \; R \; w$.

REMARK. When the modal operators are informally interpreted as denoting logical necessity and possibility the members of G are sometimes referred to as "possible worlds," whilst the binary relation R of the frame is called the "accessibility relation." We shall adopt the latter convention, but call the elements of G "points" for brevity. (END OF REMARK.)

A *first-order* frame is a 4-tuple $\left\langle G, R, D, \bar{D} \right\rangle$ where $\langle G, R \rangle$ is a frame, D is a non-empty set and $\bar{D}$ is a mapping from G to non-empty subsets of D. We

$\mathcal{L}$	Condition on R
K	no conditions
K4	transitive
D	idealisation
D4	idealisation, transitive
T	reflexive
S4	reflexive, transitive
S5	equivalence

Table 4–1: Conditions on accessibility relations.

require also that:

$$D = \bigcup_{w \in G} \bar{D}(w).$$

$\bar{D}(w)$ is interpreted as the set of individuals that "exist" at the point w.

We can obtain variants of the first-order modal logics by imposing conditions on the way $\bar{D}$ varies over its domain G. We shall be concerned in the sequel with the following three possibilities:

Varying domains: no conditions.

Cumulative domains: $\bar{D}(w) \subseteq \bar{D}(v)$, whenever $w \; R \; v$.

Constant domains: $\bar{D}(w) = \bar{D}(v)$, for all $w, v \in G$.

In this chapter we develop proof systems that are sound and complete for cumulative domains only. Similar sequent systems can be formulated for the varying domain variants of the logics [Fit83]. In §4.4 we briefly consider the problems that arise in formulating similar sequent calculi for the constant domain variants of the logics. Although the sequent methods of this chapter do not extend to all these variants, the matrix systems developed in the sequel do.

REMARKS. Since the accessibility relation for S5 is an equivalence relation, each point is either accessible from every point of the frame, including itself, or the frame partitions into "disjoint" equivalence classes. It turns out that we can always restrict attention to the first type of frame without any loss of generality [HC68].

Under the restriction mentioned above, the class of cumulative domain first-order frames for S5 reduces to the constant domain class. Basically, since every point w is accessible from every other point v, the cumulative domain condition gives us: $\bar{D}(w) \subseteq \bar{D}(v)$ and $\bar{D}(v) \subseteq \bar{D}(w)$; *i.e.*, $\bar{D}(w) = \bar{D}(v)$. (END OF REMARKS.)

If $\langle G, R, D, \bar{D} \rangle$ is a first-order frame, and $\|\!-$ a relation between the points of G and sentences (of the modal language over D), then $\langle G, R, D, \bar{D}, \|\!- \rangle$ is a first-order Kripke model provided: for each $w \in G$,

1. $w \|\!- A \wedge B$ iff $w \|\!- A$ and $w \|\!- B$.

2. $w \|\!- A \vee B$ iff either $w \|\!- A$ or $w \|\!- B$.

3. $w \|\!- A \Rightarrow B$ iff either $w \|\!\not- A$ or $w \|\!- B$.

4. $w \|\!- \neg A$ iff $w \|\!\not- A$.

5. $w \|\!- \forall x A$ iff for all $c \in \bar{D}(w)$, $w \|\!- A[c/x]$.

6. $w \|\!- \exists x A$ iff for some $c \in \bar{D}(w)$, $w \|\!- A[c/x]$.

7. $w \|\!- \Box A$ iff for all $v \in G$ with $w \; R \; v$, $v \|\!- A$.

8. $w \|\!- \Diamond A$ iff for some $v \in G$ with $w \; R \; v$, $v \|\!- A$.

The relation $w \|\!- A$ should be read: "A is *forced* at the point w," or "w *forces* A."

A sentence A, of the modal language over D, is said to be *valid* in the model $\langle G, R, D, \bar{D}, \|\!- \rangle$, provided, for each $w \in G$ such that the constants of A are in $\bar{D}(w)$, we have $w \|\!- A$.

This defines what it means for a sentence of the language of a model to be valid in that model. Let D_0 be another set of constants disjoint from D. An *interpretation* of a language over D_0 in the model $\langle G, R, D, \bar{D}, \Vdash \rangle$ is a mapping $\iota: D_0 \mapsto D$. A sentence, A, (of the modal language over D_0) is said to be *valid under the interpretation* ι in the model $\langle G, R, D, \bar{D}, \Vdash \rangle$, just in case $\iota(A)$ is valid in that model. Such a sentence is *valid* just in case it is valid under every interpretation in every model.

REMARKS. In fact we are interested in sentences that are valid in a restricted class of models: the S4-models or the K-models *etc.* In this respect we should speak of $\mathcal{L}$-models and $\mathcal{L}$-validity for any of the logics $\mathcal{L}$ under consideration. Most of the material applies to all models and logics. Consequently we shall continue to speak in terms of "models" and "validity," being more precise when the need arises.

Notice that the sentential connectives and quantifiers are interpreted relative to a single point in the frame. In contrast, the modal operators are interpreted relative to multiple points of the frame via the accessibility relation. In classical logic a formula can be either forced (true) or not (false) in a given model. In modal logics a formula may be forced or not at each point in a model. Each point of a modal model can be seen as a classical model in its own right. (END OF REMARKS.)

4.2.3 Uniform notation.

Once again we shall make substantial use of a uniform notation for formula occurrences to reduce the number of cases we need to consider in the metatheory. The notation is an extension of that introduced in Part I for classical logic, and is a minor modification of that introduced by Fitting [Fit72]. It takes advantage of the symmetry between the modal operators. We assume that the formulae in what follows are of the modal language over an arbitrary, but fixed, set of constants D.

α	α_1	α_2
$\langle A \wedge B, 1\rangle$	$\langle A, 1\rangle$	$\langle B, 1\rangle$
$\langle A \vee B, 0\rangle$	$\langle A, 0\rangle$	$\langle B, 0\rangle$
$\langle A \Rightarrow B, 0\rangle$	$\langle A, 1\rangle$	$\langle B, 0\rangle$
$\langle \neg A, 1\rangle$	$\langle A, 0\rangle$	$\langle A, 0\rangle$
$\langle \neg A, 0\rangle$	$\langle A, 1\rangle$	$\langle A, 1\rangle$

γ	$\gamma_0(a)$
$\langle \forall x A, 1\rangle$	$\langle A[a/x], 1\rangle$
$\langle \exists x A, 0\rangle$	$\langle A[a/x], 0\rangle$

ν	ν_0
$\langle \Box A, 1\rangle$	$\langle A, 1\rangle$
$\langle \Diamond A, 0\rangle$	$\langle A, 0\rangle$

β	β_1	β_2
$\langle A \wedge B, 0\rangle$	$\langle A, 0\rangle$	$\langle B, 0\rangle$
$\langle A \vee B, 1\rangle$	$\langle A, 1\rangle$	$\langle B, 1\rangle$
$\langle A \Rightarrow B, 1\rangle$	$\langle A, 0\rangle$	$\langle B, 1\rangle$

δ	$\delta_0(a)$
$\langle \forall x A, 0\rangle$	$\langle A[a/x], 0\rangle$
$\langle \exists x A, 1\rangle$	$\langle A[a/x], 1\rangle$

π	π_0
$\langle \Box A, 0\rangle$	$\langle A, 0\rangle$
$\langle \Diamond A, 1\rangle$	$\langle A, 1\rangle$

Table 4–2: Uniform notation for signed modal formulae.

A *signed* modal formula is a pair: $\langle A, n\rangle$, where A is a formula and $n \in \{0, 1\}$. We let X, Y, Z, possibly subscripted, range over signed modal formulae. A signed formula is said to be atomic if its constituent formula is atomic; otherwise it is non-atomic. We also speak of the major symbol of a signed formula by which we mean the major symbol of its component formula.

Non-atomic signed formulae whose major symbol is a sentential connective or quantifier are classified (as before) as either α, β, γ or δ type depending on their sign. Two new classes, ν and π, are required for signed formulae whose major symbol is a modal operator. Table 4–2 contains the complete classification.

IMPORTANT NOTATIONAL POINT. We shall abuse our notation extensively and use α, α_1, α_2, β, β_1, β_2, γ, $\gamma_0(a)$, ..., *etc*, to denote signed formulae of the respective types and their immediate subformulae. Consequently, when we refer to a formula α, we mean a modal formula of type α. Similarly for the

other types. This abuse of notation is used extensively by Smullyan [Smu68] and Fitting [Fit83]. (END OF POINT.)

The semantic notions of the previous section can be extended to signed formulae. For a model $\langle G, R, D, \bar{D}, \Vdash \rangle$, and $w \in G$, we define:

$$w \Vdash \langle A, 1 \rangle \quad \text{iff} \quad w \Vdash A$$

and

$$w \Vdash \langle A, 0 \rangle \quad \text{iff} \quad w \nVdash A.$$

From these definitions we obtain the following corollary:

COROLLARY 4.1 *Let* $\langle G, R, D, \bar{D}, \Vdash \rangle$ *be a model. For every* $w \in G$,

1. *Exactly one of* $w \Vdash \langle A, 1 \rangle$ *or* $w \Vdash \langle A, 0 \rangle$.
2. $w \Vdash \alpha$ *iff* $w \Vdash \alpha_1$ *and* $w \Vdash \alpha_2$.
3. $w \Vdash \beta$ *iff* $w \Vdash \beta_1$ *or* $w \Vdash \beta_2$.
4. $w \Vdash \gamma$ *iff for each* $c \in \bar{D}(w)$, $w \Vdash \gamma_0(c)$.
5. $w \Vdash \delta$ *iff for some* $c \in \bar{D}(w)$, $w \Vdash \delta_0(c)$.
6. $w \Vdash \nu$ *iff for all* $v \in G$ *such that* $w \; R \; v$, $v \Vdash \nu_0$.
7. $w \Vdash \pi$ *iff for some* $v \in G$ *such that* $w \; R \; v$, $v \Vdash \pi_0$.

A set S, of signed (or unsigned) sentences over D, is said to be forced at a point w in a model, just in case every element of the set is forced at that point. We denote this situation by $w \Vdash S$.

A set, S, of signed (or unsigned) sentences of the modal language over D_0, is *satisfiable* if there is some model $\langle G, R, D, \bar{D}, \Vdash \rangle$, and some interpretation $\iota: D_0 \mapsto D$ such that, for some point $w \in G$, all constants of $\iota(S)$ are in $\bar{D}(w)$ and $w \Vdash \iota(S)$. S is said to be *unsatisfiable* otherwise.

4.2.4 Sequents and a language for proofs.

For the proof theory, we consider a modal language as before, but with the set of constants comprising:

- A denumerable set of individual constants.
- A denumerable set of parameters.

Henceforth, we call the union of these sets D_0. We use a, b to denote parameters in D_0, and c, d to denote constants or parameters in D_0.

A sequent is an ordered pair $\langle \Gamma, \Delta \rangle$ of sets of sentences, written $\Gamma \longrightarrow \Delta$. Γ is the *antecedent* and Δ the *succedent* of the sequent; we write $\longrightarrow \Delta$ and $\Gamma \longrightarrow$ for the sequents $\langle \emptyset, \Delta \rangle$ and $\langle \Gamma, \emptyset \rangle$ respectively. Following convention, we write Γ, A for the set $\Gamma \cup \{A\}$.

A sequent $\Gamma \longrightarrow \Delta$ over D is *valid in a model* $\left\langle G, R, D, \bar{D}, \Vdash \right\rangle$, just in case: for all $w \in G$, such that the constants of the sequent are in $\bar{D}(w)$, if $w \Vdash \Gamma$ then, for some $A \in \Delta$, $w \Vdash A$. A sequent $\Gamma \longrightarrow \Delta$ over D_0 is *valid under the interpetation* $\iota: D_0 \mapsto D$, in the model $\left\langle G, R, D, \bar{D}, \Vdash \right\rangle$, just in case $\iota(\Gamma) \longrightarrow \iota(\Delta)$ is valid in the model. Finally, a sequent is *valid* if and only if it is valid under every interpretation in every model.

Recall that that a sequent $\Gamma \longrightarrow \Delta$ determines a set $S_\Gamma \cup S_\Delta$ of signed formulae where:

$$S_\Gamma \stackrel{\text{df}}{=} \{ \langle A, 1 \rangle \mid A \in \Gamma \}$$
$$S_\Delta \stackrel{\text{df}}{=} \{ \langle A, 0 \rangle \mid A \in \Delta \}.$$

This set is called the *associated* set of the sequent.

The following proposition summarises the semantic relationship between sequents and their associated sets.

PROPOSITION 4.2 *A sequent* $\Gamma \longrightarrow \Delta$ *is valid if and only if its associated set is unsatisfiable.*

PROOF. The associated set is unsatisfiable just in case for every model: $\langle G, R, D, \bar{D}, \Vdash \rangle$, every interpretation ι of the constants of the set in the model, and every $w \in G$ such that these constants are in $\bar{D}(w)$, it is not the case that $w \Vdash \iota(S_\Gamma \cup S_\Delta)$.

Now, $w \Vdash \iota(S_\Gamma \cup S_\Delta)$ iff both $w \Vdash \iota(S_\Gamma)$ and $w \Vdash \iota(S_\Delta)$. So $w \Vdash \iota(S_\Gamma \cup S_\Delta)$ fails just when the implication: "if $w \Vdash \iota(S_\Gamma)$, then it is not the case that $w \Vdash \iota(S_\Delta)$" holds. But:

$$w \Vdash \iota(S_\Gamma) \quad \text{iff} \quad w \Vdash \iota(\Gamma)$$

since every $X \in S_\Gamma$ has $X = \langle B, 1 \rangle$, for some formula B. Similarly,

$$w \Vdash \iota(S_\Delta) \text{ fails} \quad \text{iff} \quad w \nVdash \iota(Y)$$

for some $Y \in S_\Delta$. But every such Y has $Y = \langle A, 0 \rangle$ for some $A \in \Delta$. Therefore,

$$w \Vdash \iota(S_\Delta) \text{ fails} \quad \text{iff} \quad w \Vdash \iota(A)$$

for some $A \in \Delta$. To summarise: the negation of $w \Vdash \iota(S_\Gamma \cup S_\Delta)$ is equivalent to the implication: "if $w \Vdash \iota(\Gamma)$, then for some $A \in \Delta$, $w \Vdash \iota(A)$." But this is just a statement of the validity of $\Gamma \longrightarrow \Delta$ under ι in the model $\langle G, R, D, \bar{D}, \Vdash \rangle$. ■

4.3 Sequent calculi for modal logics.

In this section we develop cut-free sequent calculi for the modal logics under consideration. The material is reasonably standard and has been adapted from [Fit83]. In Chapter 5 we discuss the suitability of these calculi as bases for automated proof search. As the reader might expect in the light of previous chapters, where we discussed such issues for the classical sequent calculus, the systems are found wanting. An analysis of the particular redundancies introduced by the rules for the modal operators motivates the ensuing development of the more efficient matrix systems in Chapter 6.

All of the sequent calculi for the modal logics are essentially variants on a basic theme. We exploit this in our presentation. First, we develop the basic calculus for S4 in §4.3.1. The calculi for the other logics are presented as variants of this calculus in §4.3.2. We include correctness results and examples to give the reader a flavour of the arguments to follow in Chapters 5 and 6.

4.3.1 A sequent calculus for S4.

In this section we develop (§4.3.1.1), and prove correct (§4.3.1.2), a cut-free sequent calculus for S4. Recall that the accessibility relation for S4-models is constrained to be both reflexive and transitive. The proof system is designed to determine the valid sequents of the logic.

4.3.1.1 The calculus.

As usual with cut-free sequent calculi, the rules fall into three categories: *basic sequents* or axioms, *operational* rules and *structural* rules. Since our sequents are sets of formulae rather than sequences, we have no need for structural rules.

The basic sequents are instances of the schema:

$$\Gamma, A \longrightarrow A, \Delta .$$

This is identical to the schema defining the basic sequents for the classical sequent calculus of Part I.

The operational rules appear in pairs, each pair associated with a particular sentential connective, quantifier or modal operator. One rule introduces the connective/quantifier/operator into the antecedent, the other introduces it into the succedent. Since classical propositional logic forms a fragment of the modal logics, the rules for the former are a subset of the rules for the latter. The quantifier rules are also lifted directly from the classical calculus. We need additional rules for manipulating the modal operators. Recall that these operators, unlike the sentential connectives and quantifiers, are interpreted relative to more

than one point in a model. The manner in which this is captured by the modal rules will become clear in the correctness proofs below. The complete system is summarised in Figure 4–1.

Derivations are defined (as usual) as coherent trees of rule instances. We omit the details. *Proofs* are derivations whose leaves are instances of the basic sequent.

4.3.1.2 Correctness and completeness.

We now justify the rules by showing that they express theorems about valid sequents. The form of these theorems is as follows. For each rule:

> if all the premises are valid, then the conclusion is valid.

In particular the basic sequent (with no premises) is valid. To simplify the arguments we employ the uniform notation introduced above.

In what follows, by: "satisfiable," "model," "valid," *etc*, we mean: "S4-satisfiable," "S4-model," "S4-valid," *etc*, repectively.

Recall that a sequent determines an associated set of signed formulae. The rules of Figure 4–1 can be rephrased concisely in terms of these associated sets. We shall motivate the form of the rephrased rules with reference to the modal rule:

$$\frac{\Gamma,\ A \longrightarrow \Delta}{\Gamma,\ \Box A \longrightarrow \Delta}\ \Box \longrightarrow$$

If S_Γ and S_Δ are the sets of signed formulae associated with Γ and Δ respectively, *i.e.*,

$$S_\Gamma \stackrel{\text{df}}{=} \{\, \langle B, 1\rangle \mid B \in \Gamma \,\}$$
$$S_\Delta \stackrel{\text{df}}{=} \{\, \langle B, 0\rangle \mid B \in \Delta \,\},$$

then the set associated with the premise is:

$$S_\Gamma, S_\Delta, \langle A, 1\rangle.$$

$$\Gamma, A \longrightarrow A, \Delta$$

$$\frac{\Gamma \longrightarrow A, \Delta \quad \Gamma, B \longrightarrow \Delta}{\Gamma, A \Rightarrow B \longrightarrow \Delta} \Rightarrow\!\longrightarrow \qquad \frac{\Gamma, A \longrightarrow B, \Delta}{\Gamma \longrightarrow A \Rightarrow B, \Delta} \longrightarrow\!\Rightarrow$$

$$\frac{\Gamma, A, B \longrightarrow \Delta}{\Gamma, A \wedge B \longrightarrow \Delta} \wedge\!\longrightarrow \qquad \frac{\Gamma \longrightarrow A, \Delta \quad \Gamma \longrightarrow B, \Delta}{\Gamma \longrightarrow A \wedge B, \Delta} \longrightarrow\!\wedge$$

$$\frac{\Gamma, A \longrightarrow \Delta \quad \Gamma, B \longrightarrow \Delta}{\Gamma, A \vee B \longrightarrow \Delta} \vee\!\longrightarrow \qquad \frac{\Gamma \longrightarrow A, B, \Delta}{\Gamma \longrightarrow A \vee B, \Delta} \longrightarrow\!\vee$$

$$\frac{\Gamma \longrightarrow A, \Delta}{\Gamma, \neg A \longrightarrow \Delta} \neg\!\longrightarrow \qquad \frac{\Gamma, A \longrightarrow \Delta}{\Gamma \longrightarrow \neg A, \Delta} \longrightarrow\!\neg$$

$$\frac{\Gamma, A[c/x] \longrightarrow \Delta}{\Gamma, \forall x A \longrightarrow \Delta} \forall\!\longrightarrow \qquad \frac{\Gamma \longrightarrow A[a/x], \Delta}{\Gamma \longrightarrow \forall x A, \Delta} \longrightarrow\!\forall$$

$$\frac{\Gamma, A[a/x] \longrightarrow \Delta}{\Gamma, \exists x A \longrightarrow \Delta} \exists\!\longrightarrow \qquad \frac{\Gamma \longrightarrow A[c/x], \Delta}{\Gamma \longrightarrow \exists x A, \Delta} \longrightarrow\!\exists$$

$$\frac{\Gamma, A \longrightarrow \Delta}{\Gamma, \Box A \longrightarrow \Delta} \Box\!\longrightarrow \qquad \frac{\Gamma* \longrightarrow A, \Delta*}{\Gamma \longrightarrow \Box A, \Delta} \longrightarrow\!\Box$$

$$\frac{\Gamma*, A \longrightarrow \Delta*}{\Gamma, \Diamond A \longrightarrow \Delta} \Diamond\!\longrightarrow \qquad \frac{\Gamma \longrightarrow A, \Delta}{\Gamma \longrightarrow \Diamond A, \Delta} \longrightarrow\!\Diamond$$

For the $\longrightarrow \forall$ and $\exists \longrightarrow$ rules: the parameter a must not occur in the conclusion.

$$\Gamma* \stackrel{\text{df}}{=} \{ \Box B \mid \Box B \in \Gamma \}$$
$$\Delta* \stackrel{\text{df}}{=} \{ \Diamond B \mid \Diamond B \in \Delta \}$$

Figure 4–1: A cut-free sequent calculus for S4.

Similarly the set of signed formulae associated with the conclusion of the rule can be expressed as follows:

$$S_\Gamma, S_\Delta, \langle \Box A, 1 \rangle.$$

But the signed formula $\langle \Box A, 1 \rangle$ is of ν-type according to Table 4–2, consequently the rule can be written:

$$\frac{S, \nu_0}{S, \nu}$$

where S is the union of S_Γ and S_Δ. By a similar analysis all four modal rules collapse to the pair:

$$\frac{S, \nu_0}{S, \nu} \ \nu \qquad \frac{S*, \pi_0}{S, \pi} \ \pi$$

where

$$S* \stackrel{\text{df}}{=} \{ \nu \mid \nu \in S \}.$$

(Recall that ν-type formulae are those of the form: $\langle \Box B, 1 \rangle$ and $\langle \Diamond B, 0 \rangle$ for some formula B. Consequently, the notation $\{ \nu \mid \nu \in S \}$ means the set of ν-type formula that are elements of S.) The eight sentential operational rules can be expressed uniformly as the following pair:

$$\frac{S, \alpha_1, \alpha_2}{S, \alpha} \ \alpha \qquad \frac{S, \beta_1 \quad S, \beta_2}{S, \beta} \ \beta.$$

The basic sequent becomes simply:

$$S, \langle A, 1 \rangle, \langle A, 0 \rangle.$$

The condensed system is summarised in Figure 4–2. This leaves us with only seven correctness theorems to prove instead of seventeen.

LEMMA 4.3 *A set of signed formulae of the form* $S, \langle A, 1 \rangle, \langle A, 0 \rangle$ *is unsatisfiable.*

PROOF. By case (1), Corollary 4.1, $\langle A, 1 \rangle$ and $\langle A, 0 \rangle$ cannot both be forced at the same point of a model. ■

PROPOSITION 4.4 *All basic sequents are valid.*

$$S, \langle A, 1\rangle, \langle A, 0\rangle$$

$$\frac{S, \alpha_1, \alpha_2}{S, \alpha}\ \alpha \qquad \frac{S, \beta_1 \quad S, \beta_2}{S, \beta}\ \beta$$

$$\frac{S, \gamma_0(a)}{S, \gamma}\ \gamma \qquad \frac{S, \delta_0(a)}{S, \delta}\ \delta$$

$$\frac{S, \nu_0}{S, \nu}\ \nu \qquad \frac{S*, \pi_0}{S, \pi}\ \pi$$

For the δ rule, the parameter a must not occur in the conclusion.

$$S* \stackrel{\text{df}}{=} \{\, \nu \mid \nu \in S \,\}$$

Figure 4–2: A Cut-free sequent calculus for S4 in uniform notation.

PROOF. Follows immediately from the above lemma and Proposition 4.2. ■

LEMMA 4.5 *If S, α is satisfiable, so is S, α_1, α_2.*

PROOF. S, α satisfiable implies there is some model $\langle G, R, D, \bar{D}, \Vdash \rangle$, interpretation $\iota: D_0 \mapsto D$, and $w \in G$ with all constants of $\iota(S)$ and $\iota(\alpha)$ in $\bar{D}(w)$, such that: $w \Vdash \iota(S)$ and $w \Vdash \iota(\alpha)$. But then, by case (2), Corollary 4.1, $w \Vdash \iota(\alpha_i)$, $i = 1, 2$. Hence S, α_1, α_2 is satisfiable at the same point, in the same model, under the same interpretation. ■

PROPOSITION 4.6 *The sentential rules with one premise (i.e., $\wedge \longrightarrow$, $\neg \longrightarrow$, $\longrightarrow \neg$, $\longrightarrow \vee$, $\longrightarrow \Rightarrow$), are correct.*

We prove this result in detail. Similar proofs in the sequel will be shortened or omitted.

PROOF. The associated sets of the premise and conclusion of each of these rules have the form S, α_1, α_2 and S, α respectively. The contrapositive of Lemma 4.5 reads: if S, α_1, α_2 is unsatisfiable, so is S, α. Hence, by Proposition 4.2, if the premise of such a sequent rule is valid, so is the conclusion, *i.e.*, it is correct. ■

LEMMA 4.7 *If S, β is satisfiable, so is at least one of S, β_1 or S, β_2.*

PROOF. Again, let the model $\langle G, R, D, \bar{D}, \Vdash \rangle$ satisfy S, β under an interpretation ι, at $w \in G$. By case (3), Corollary 4.1, $w \Vdash \iota(\beta)$ iff $w \Vdash \iota(\beta_1)$ or $w \Vdash \iota(\beta_2)$. The result follows immediately. ■

PROPOSITION 4.8 *The sentential rules with two premises (i.e., $\longrightarrow \wedge$, $\vee \longrightarrow$, $\Rightarrow \longrightarrow$), are correct.*

LEMMA 4.9 *If S, γ is satisfiable, so is $S, \gamma_0(a)$, for some constant or parameter $c \in D_0$.*

PROOF. Let the model $\langle G, R, D, \bar{D}, \Vdash \rangle$ satisfy S, γ at $w \in G$, under an interpretation ι. Then $\gamma_0(c)$ (possibly) contains c. There are two cases depending on whether c is new, or is already a constant of S, γ.

1. c is new. Then ι is not defined on c. Choose a $c' \in \bar{D}(w)$ (which is non-empty by definition). Define ι' on the constants of $S, \gamma_0(c)$ as follows:

$$\iota'(d) = \begin{cases} c', & d = c \\ \iota(d), & \text{otherwise.} \end{cases}$$

   $w \Vdash \iota'(S)$ by hypothesis, and the fact that ι' agrees with ι on the constants of S. We have:

$$\iota'(\gamma_0(c)) = \iota'(\gamma_0)(\iota'(c)) = \iota'(\gamma_0)(c').$$

   since $\iota'(c) = c'$. Moreover, by case (5), Corollary 4.1,

$$w \Vdash \iota'(\gamma) \quad \text{iff} \quad w \Vdash \iota'(\gamma_0)(d)$$

   for every $d \in \bar{D}(w)$. But $c' \in \bar{D}(w)$ by construction. Hence:

$$w \Vdash \iota'(\gamma_0(c))$$

   and $S, \gamma_0(c)$ is satisfiable.

2. c is not new. Then ι is already defined on c and $\iota(c) \in \bar{D}(w)$ by hypothesis. The model condition cited above ensures that $w \Vdash \iota(\gamma_0(c))$.

■

PROPOSITION 4.10 *The quantifier rules* $\forall \longrightarrow$ *and* $\longrightarrow \exists$ *are correct.*

LEMMA 4.11 *If* S, δ *is satisfiable, so is* $S, \delta_0(a)$, *for some parameter* $a \in D_0$ *which does not occur in* S, δ.

PROOF. Let the model $\langle G, R, D, \bar{D}, \Vdash \rangle$ satisfy S, δ at $w \in G$, under an interpretation ι. ι is not defined on a, since a does not occur in S, δ. By case (6), Corollary 4.1,

$$w \Vdash \iota(\delta) \quad \text{iff} \quad w \Vdash \iota(\delta_0)(c)$$

for some $c \in \bar{D}(w)$. Extend ι as follows:

$$\iota'(d) = \begin{cases} c, & d = a \\ \iota(d), & \text{otherwise.} \end{cases}$$

$w \Vdash \iota'(S)$ by hypothesis, and the fact that ι' agrees with ι on the constants of S. We have:

$$\iota'(\delta_0(a)) = \iota'(\delta_0)(\iota'(a)) = \iota'(\delta_0)(c).$$

since $\iota'(a) = c$. Furthermore, by construction, $c \in \bar{D}(w)$ and:

$$w \Vdash \iota'(\delta_0)(c).$$

Hence

$$w \Vdash \iota'(\delta_0(a)),$$

and $S, \delta_0(a)$ is satisfiable. ■

PROPOSITION 4.12 *The sequent rules* $\exists \longrightarrow$ *and* $\longrightarrow \forall$ *are correct.*

The previous five propositions, dealing with the classical parts of the proof system, do not rely on a particular modal logic. The final two propositions that deal with the modal rules are logic-dependent. They also indicate why the sequent systems presented are correct only with respect to the class of cumulative domain S4-models.

LEMMA 4.13 *If* S, ν *is S4-satisfiable, so is* S, ν_0.

PROOF. Let the model $\langle G, R, D, \bar{D}, \Vdash \rangle$ satisfy S, ν at $w \in G$, under an interpretation ι. The constants of $\iota(S)$ and $\iota(\nu_0)$ are in $\bar{D}(w)$ by hypothesis. Since R is reflexive, $w\ R\ w$ holds. Consequently, $w \Vdash \iota(\nu_0)$, by case (6), Corollary 4.1. Hence S, ν_0 is satisfiable at w also. ■

Notice that the proof relies on the fact that the accessibility relation for S4 is reflexive.

PROPOSITION 4.14 *The rules* $\Box \longrightarrow$ *and* $\longrightarrow \Diamond$ *are correct.*

LEMMA 4.15 *If S, π is S4-satisfiable, so is $S*, \pi_0$.*

PROOF. Recall that $S*$ is the set $\{ \nu \mid \nu \in S \}$. Suppose S, π is satisfied in the model: $\langle G, R, D, \bar{D}, \Vdash \rangle$, at $w \in G$, under an interpretation ι. $w \Vdash \iota(\pi)$ implies that there is a $v \in G$, with $w \; R \; v$ such that $v \Vdash \iota(\pi_0)$, by case (7), Corollary 4.1. We claim that $v \Vdash \iota(S*)$, and hence the result.

To prove the claim, consider a $\nu \in S*$ and $u \in G$ with $v \; R \; u$. We know that $w \Vdash \iota(\nu)$. Since R is transitive, $w \; R \; u$. Consequently $u \Vdash \iota(\nu_0)$, by case (6), Corollary 4.1. But u is arbitrary, hence for every u with $v \; R \; u$, $u \Vdash \iota(\nu_0)$, *i.e.*, $v \Vdash \iota(\nu)$ by case (6), Corollary 4.1 once again. Since ν was arbitrary, the claim follows. ■

Notice that the proof relies on the fact that the accessibility relation for S4 is transitive.

PROPOSITION 4.16 *The rules $\longrightarrow \Box$ and $\Diamond \longrightarrow$ are correct.*

REMARK. Notice that the point at which the premise of the π rule is satisfied is, in general, distinct from the point at which the conclusion is satisfied in a model; the former being accessible from the latter. The model condition for δ formulae sanctions the inference of $w \Vdash \delta_0(c)$ from $w \Vdash \delta$, for some $c \in \bar{D}(w)$. Proof-theoretically this is represented by the use of a parameter, a, in the δ rule. a is interpreted as denoting this particular constant $c \in \bar{D}(w)$ of the model. Now assume that our attention changes, by means of the π modal rule to another point w' of the model, with $w \; R \; w'$. The model condition for γ formulae sanctions the inference of $w' \Vdash \gamma_0(c)$ from $w' \Vdash \gamma$, for any $c \in \bar{D}(w')$. The γ rule, however, allows the inference $\gamma_0(a)$ from γ for *any* parameter or constant $a \in D_0$. In particular, parameters introduced by the δ rule at the previous point w are assumed to have an interpretation at the new (but accessible) point w'. This will only be correct if the interpretation of the parameter, $c \in \bar{D}(w)$, asserted to exist at w, is guaranteed to exist at the accessible point w'. The combination of the liberal quantifier rules and the modal rules is thus only correct for the

cumulative domain models where by definition: $\bar{D}(w) \subseteq \bar{D}(w')$. (END OF REMARK.)

The above lemmata serve to establish the correctness of the system.

THEOREM 4.17 (CORRECTNESS) *The proof system summarised in Figure 4-1 is correct for the cumulative domain variant of S4; i.e., any sequent provable via the system is S4-valid.*

The systematic techniques of Kleene [Kle68], Smullyan [Smu68] or Fitting [Fit83] can be adapted to demonstrate the completeness of the calculus presented. The proof relies on an adaptation of König's Lemma for finitely branching infinite trees. The method is to systematically construct a derivation for the endsequent $\longrightarrow A$, starting from the root, such that either:

(a) The procedure terminates with a proof.

(b) The procedure terminates with one of the leaves of the derivation not being an instance of the basic sequent.

(c) The procedure continues for ever.

The construction is performed in such a way that if termination occurs as in (b), or the procedure fails to terminate as in (c), enough information has been produced to form a model in which A is not valid. Completeness follows immediately.

REMARK. This form of systematic completeness proof also furnishes us with a semi-decision procedure for the undecidable quantified logics, and can be modified to produce decision procedures for the propositional fragments. Unfortunately, as we will show in the next chapter, such procedures whilst technically elegant are less than adequate for automated proof search. (END OF REMARK.)

EXAMPLE. We conclude this section on S4 with a couple of example proofs using the proof system developed above.

Our first example is a proof of the (schematic) sentence: $\Box A \Rightarrow \Box\Box A$, where A is some arbitrary formula. This is usually taken to be the defining axiom for S4 in axiomatic presentations such as [HC68].

$$\dfrac{\dfrac{\dfrac{\dfrac{A \longrightarrow A}{\Box A \longrightarrow A}\,(\Box\longrightarrow)}{\Box A \longrightarrow \Box A}\,(\longrightarrow\Box)}{\Box A \longrightarrow \Box\Box A}\,(\longrightarrow\Box)}{\longrightarrow \Box A \Rightarrow \Box\Box A}\,(\longrightarrow\Rightarrow)$$

We have included the classification of each rule application for the reader's convenience. Notice, how the formula $\Box A$ in the antecedent is "preserved" by successive $\longrightarrow \Box$ applications. The reader should consider how the proof would go wrong if the $\Box \longrightarrow$ rule was applied first. We shall examine such problems in detail in the next chapter. (END OF EXAMPLE.)

EXAMPLE. Our second example involves the use of β rules as well as both types of modal rule. We prove a modal form of modus ponens: $\Box A \wedge \Box(A \Rightarrow B) \Rightarrow \Box B$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A \longrightarrow A, B}{\Box A \longrightarrow A, B}\,(\Box\longrightarrow) \quad \Box A, B \longrightarrow B}{\Box A, A \Rightarrow B \longrightarrow B}\,(\Rightarrow\longrightarrow)}{\Box A, \Box(A \Rightarrow B) \longrightarrow B}\,(\Box\longrightarrow)}{\Box A, \Box(A \Rightarrow B) \longrightarrow \Box B}\,(\longrightarrow\Box)}{\Box A \wedge \Box(A \Rightarrow B) \longrightarrow \Box B}\,(\wedge\longrightarrow)}{\longrightarrow \Box A \wedge \Box(A \Rightarrow B) \Rightarrow \Box B}\,(\longrightarrow\Rightarrow)$$

(END OF EXAMPLE.)

4.3.2 Sequent calculi for K, K4, D, D4, T, S4.

Above, we developed a sequent calculus for S4. Here we indicate how that system can be modified to obtain correct and complete sequent calculi for the cumulative domain variants of all of the modal logics under consideration.

Classical logic is a fragment of each of the modal logics. This is reflected in the fact that the basic sequents, and the α, β, γ and δ rules are taken directly

from the sequent system given for classical logic presented in Part I. The reader should check that the correctness proofs for these rules do not rely on any specific properties of S4-models over and above those shared by classical models (the latter corresponding to a point of the former). We conclude that these rules are correct for every (cumulative domain) modal logic.

This is not the case for the ν and π rules. We remarked at the time that the correctness (Lemma 4.13) of the ν rules ($\Box \longrightarrow$ and $\longrightarrow \Diamond$) relies on the reflexivity of the accessibility relation for the logic. It is, therefore, only correct for the logics T and S4. Likewise, the correctness proof (Lemma 4.15) for the π rules ($\Diamond \longrightarrow$ and $\longrightarrow \Box$) relies on the transitivity of the accessibility relation for the logic. Therefore, this rule is only correct for the logics K4, D4 and S4.

The situation is summarised in the following lemma:

LEMMA 4.18 *Let* $\langle G, R, D, \bar{D}, \|- \rangle$ *be an* $\mathcal{L}$*-model,* S *a set of signed formulae over* D*, and* $w, v \in G$ *such that* $w\ R\ v$*. Then,*

1. $w \|- S$ *implies* $v \|- \{ \nu_0 \mid \nu \in S \}$.
2. *If* R *is transitive,* $w \|- S$ *implies* $v \|- \{ \nu \mid \nu \in S \}$.
3. *If* R *is reflexive,* $w \|- \nu$ *implies* $w \|- \nu_0$.

PROOF. The first assertion follows immediately from case (6), Corollary 4.1. The second was proved as part of the proof of Lemma 4.15 above. The third follows again from case (6), Corollary 4.1, together with the fact that $w\ R\ w$ since R is reflexive, and was proved as part of Lemma 4.13 above. ■

Finally we have the distinction between the logics whose models satisfy the idealisation condition, and those whose models do not, namely K and K4. The idealisation condition states that for every point in a model, there is another point accessible from it. For the K-logics this is not the case. This manifests itself proof-theoretically in the absence of any ν rule whatsoever for the K-logics.

We obtain the flexibility needed to capture the logics proof-theoretically by changing the form of the ν and π rules. The definition of these rules is shown in Table 4–3.

To summarise: A sequent calculus for one of the logics K, K4, D, D4, T, S4 consists of the basic sequent rule, the α, β, γ and δ rules, and the ν and π rules specified for that logic in Table 4–3. We shall not prove the correctness nor completeness of these systems. Such proofs, for tableau proof systems, which are notational variants of the above sequent calculi, can be found in Fitting's book [Fit83]. We conclude with some examples.

EXAMPLE. As an example of the difference between some of the systems described above, we reprove the "modus ponens" example in the system for K.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A, B \longrightarrow B \qquad A \longrightarrow A, B}{A, A \Rightarrow B \longrightarrow B}\ \Rightarrow\!\longrightarrow\ (\beta)}{\Box A, \Box(A \Rightarrow B) \longrightarrow \Box B}\ \longrightarrow \Box\ (\pi)}{\Box A \wedge \Box(A \Rightarrow B) \longrightarrow \Box B}\ \wedge\longrightarrow\ (\alpha)}{\longrightarrow \Box A \wedge \Box(A \Rightarrow B) \Rightarrow \Box B}\ \longrightarrow\!\Rightarrow\ (\alpha)$$

Notice that the form of the π rule in the K system forces the modal operator to quantify every formula in the antecedent of its conclusion. We expect then that implicational sentences with different numbers of similar modalities quantifying its conclusion and antecedent will not be provable in this system. The S4-axiom $\Box A \Rightarrow \Box\Box A$ is one such formula.

$$\dfrac{\dfrac{\dfrac{\dfrac{?}{\longrightarrow A}}{A \longrightarrow \Box A}}{\Box A \longrightarrow \Box\Box A}}{\longrightarrow \Box A \Rightarrow \Box\Box A}\quad\begin{array}{ll} ? & (?)\\ \longrightarrow \Box & (\pi)\\ \longrightarrow \Box & (\pi)\\ \longrightarrow\!\Rightarrow & (\alpha)\end{array}\qquad\qquad \dfrac{\dfrac{\dfrac{A \longrightarrow A}{\Box A \longrightarrow \Box A}}{\Box\Box A \longrightarrow \Box\Box A}}{?}\quad\begin{array}{ll} \longrightarrow \Box & (\pi)\\ \longrightarrow \Box & (\pi)\\ ? & (?)\end{array}$$

The left-hand K-derivation above indicates that $\Box A \Rightarrow \Box\Box A$ is not valid in a K-model with three points: w_0, w_1 and w_2; $w_0\ R\ w_1$ and $w_1\ R\ w_2$; and $w_2 \Vdash\!\!\!/\ A$, $w_1 \Vdash A$ and $w_0 \Vdash A$. (END OF EXAMPLE.)

Logic	$S*$	ν rule	π rule
K	$\{\nu_0 \mid \nu \in S\}$	none	$\frac{S*, \pi_0}{S, \pi}$
K4	$\{\nu_0 \mid \nu \in S\} \cup \{\nu \mid \nu \in S\}$	none	$\frac{S*, \pi_0}{S, \pi}$
D	$\{\nu_0 \mid \nu \in S\}$	$\frac{S*}{S}$	$\frac{S*, \pi_0}{S, \pi}$
D4	$\{\nu_0 \mid \nu \in S\} \cup \{\nu \mid \nu \in S\}$	$\frac{S*}{S}$	$\frac{S*, \pi_0}{S, \pi}$
T	$\{\nu_0 \mid \nu \in S\}$	$\frac{S, \nu_0}{S, \nu}$	$\frac{S*, \pi_0}{S, \pi}$
S4	$\{\nu \mid \nu \in S\}$	$\frac{S, \nu_0}{S, \nu}$	$\frac{S*, \pi_0}{S, \pi}$

Table 4–3: Summary of ν and π rules for the modal logics.

EXAMPLE. We prove the sentence $\Box\forall x(Ax) \Rightarrow \forall y(\Box Ay)$ in the system for K.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Aa \longrightarrow Aa}{\forall x(Ax) \longrightarrow Aa}\,(\forall\longrightarrow)}{\Box\forall x(Ax) \longrightarrow \Box Aa}\,(\longrightarrow\Box)}{\Box\forall x(Ax) \longrightarrow \forall y(\Box Ay)}\,(\longrightarrow\forall)}{\longrightarrow \Box\forall x(Ax) \Rightarrow \forall y(\Box Ay)}\,(\longrightarrow\Rightarrow)}$$

(END OF EXAMPLE.)

4.4 S5 and constant domain modal logics.

In [Fin79] Fine shows that Beth's Definability Theorem, and hence the Craig Interpolation Lemma, fails for the constant domain variants of the quantified modal logics that we are considering. Beth's Definability Theorem for a logic states that if a predicate is implicitly definable in a theory of the logic, then it is explicitly definable in that theory. (Roughly speaking, if P is a predicate of a theory T, and T' is the result of replacing all occurrences of P with a new predicate symbol P' of the same arity as P in the axioms of T, then P is *implicitly* definable in T if

$$T, T' \vdash \forall x_1 \cdots \forall x_n (Px_1 \cdots x_n \Rightarrow P'x_1 \cdots x_n).$$

P is *explicitly* definable in T if

$$T \vdash \forall x_1 \cdots \forall x_n (Px_1 \cdots x_n \equiv A)$$

for some formula A in the language of T, not containing P.)

Fitting [Fit83] gives proofs of the Craig Interpolation Lemma for the propositional fragments of the modal logics K, K4, D, D4, T, S4 considered above. These proofs are based on so-called *symmetric* sequent calculi for the logics. The notion of a symmetric sequent calculus is easy to grasp. Certain of sentential sequent rules, namely the rules for negation and implication, cause formulae to "move" from the antecedent to the succedent and vice versa. (More precisely, a

succedent formula in the premise may end up as a subformula of an antecedent formula in the conclusion of the rule, or vice versa, when the rule is applied forwards.) Symmetric sequent calculi are calculi in which none of the rules have this property.

The sequent systems we have presented above are called *cut-free* because they do not contain the inference rule named "cut" by Gentzen [G69]:

$$\frac{\Gamma \longrightarrow A, \Delta \qquad \Gamma, A \longrightarrow \Delta}{\Gamma \longrightarrow \Delta} \text{ cut}$$

This rule arises in the translation of a *natural deduction* into a derivation in the classical sequent calculus of Part I. Cut-free sequent calculi, in general, possess the so-called *subformula* property: derivations are formed entirely from the subformulae of their endsequent.

It is a straightforward matter to construct symmetric sequent calculi from cut-free systems that possess the subformula principle. Fitting [Fit83] performs such constructions for the logics we have considered.

The import of these results is that we cannot hope to construct cut-free sequent calculi of the type presented in this chapter for the constant domain logics. If we could they could be used to construct symmetric sequent systems and hence prove the appropriate Interpolation Lemma. The Definability Theorem for the logic would then follow. This contradicts Fine's result.

These results extend to S5 itself since the first-order version of S5 has constant domains anyway if we assume the cumulativity condition (§4.2.2). In fact Fitting [Fit83] suggests that the problem lies with the condition of "symmetry" placed on the accessibility relation. S5 is the only logic we are considering whose accessibility relation is symmetric. The constant domain assumption is a form of symmetry. Individuals asserted to exist at newly considered points, must have existed at all other points as well.

A number of authors have developed sequent-based proof systems for S5. Kanger [Kan57] and Fitting [Fit72] utilise the notion of a *prefix*. Their sequent (or tableau) systems then work with augmented sequents comprising *prefixed* or

"spotted" formulae. We shall in fact adopt a similar method to capture S5 and the constant domain variants in the next chapter. Other authors have employed different methods to obtain cut-free sequent-style proof systems for S5. Sato's [Sat77] proof system utilises pairs of sequents, while Mints' system [Min70] is cut-free, but his rules fail to have the subformula property. We have seen in Part I how important the subformula property is from a practical point of view since it forms the basis for structure-sharing methods.

The constant domain logics can also be captured axiomatically by including, as an axiom, the so-called Barcan formula:

$$\forall x \Box Ax \Rightarrow \Box \forall x Ax.$$

Given the problem of obtaining a uniform proof-theoretic treatment of the modal logics, the question arises as to why we chose to introduce the cut-free systems that we did rather than, say, Fitting's prefixed systems which are general enough to cover all the logics? The answer to this question is partly historical and partly technical. Historically we were led to the non-classical matrix systems by a study of the sequent systems introduced in this chapter and the standard sequent calculus for intuitionistic logic. The fruits of that study are presented in the next chapter. The fact that our matrix solutions extended to the constant domain logics was somewhat fortuitous. Technically, we are driven by a desire to treat a wide range of proof systems. It is more beneficial to see the matrix systems as overcoming combinatorial problems in standard proof systems than in special purpose ones that may not generalise to other classes of logics. We hope that this decision will enable the application of our results to a wider class of logics than those considered here. We reconsider these points in Chapter 10.

4.5 Summary.

In this chapter we have presented the syntax and semantics (§4.2) of the modal logics K, K4, D, D4, T, S4 and S5. Using a uniform notation due to Smullyan [Smu68] and Fitting [Fit83], we presented standard sequent calculi for the cumulative domain variants of the quantified modal logics K, K4, D, D4, T, S4. The material is based on Fitting's treatment of similar material presented in [Fit83]. No claim is made for originality. We have briefly discussed the problems of formulating cut-free sequent calculi for S5 and the constant-domain versions of the modal logics.

Chapter 5

Proof search in modal sequent calculi.

5.1 Introduction.

The sequent calculi presented in the previous chapter for some of the modal logics under consideration are reasonably natural to use by hand, and have been used extensively in metatheoretic arguments about modal logics (*eg.*, [Fit72,Fit83, HC68]). Indeed, such uses formed the main motivation for their development in the first place.

As should be expected, these systems are not ideally suited for direct implementation and efficient automated proof search. In Part I we identified certain redundancies present in the search space generated by the sequent calculus for classical logic. These redundancies were broadly classified under the following headings:

- Notational redundancy: considerable duplication of the same information.
- Relevance: the inclusion in the search space of branches that cannot lead to a proof.
- Order dependence: the need to explore alternative branches in the search space that differ only in the order in which certain sequent rules are applied.

We treated the sequent system for classical logic as the composition of a calculus for the pure propositional fragment with a calculus for the quantificational part of the logic. The redundancies concerning notation and relevance were shown to arise essentially from the pure propositional subsystem, whilst the quantificational subsystem was shown to be responsible for the sensitivity of the search space to rule application order.

By means of this analysis we developed a characterisation of validity in classical logic based on matrices. The major components of this characterisation were paths, connections and a particular use of unification to overcome the order dependence induced by quantifiers. This development can be seen as a (theoretical) rational reconstruction of the work of Bibel [Bib81,Bib82a,Bib82c], and to some extent, Andrews and his colleagues [And81].

In this short chapter we investigate the structure of the search spaces generated by the sequent calculi for modal logics presented in the previous chapter. We view each propositional modal system as the composition of a calculus for the modal part of the logic with the pure propositional calculus. The quantified modal systems are then formed by the addition of the usual calculus for quantifiers.

Since the modal calculi contain the pure propositional calculus as a subsystem we should expect the first two types of redundancy, concerning notation and relevance, to occur as before. This is indeed the case. For completeness, we review these arguments briefly in §5.2. The reader is referred to Part I for a more detailed discussion.

In §5.3 we show how the addition of the modal rules to form the propositional modal calculi induces a form of order dependence similar to that found when the quantifier rules were added to the propositional calculus in Part I. We illustrate this problem with a simple example. Finally, in §5.4 we analyse the effect of adding quantifiers to the propositional modal systems. Since both the modal rules and the quantifier rules independently induce an order dependence in the search space, the question arises as to how these dependencies interact. Even though the sequent systems we have presented do not extend to S5 or the

constant domain variants of the logics, we are led to conjecture that removing the combined order-related redundancies will turn out to be simpler in these cases than in the standard quantified modal logics with (arbitrarily) varying or cumulative domains.

In the next chapter we develop matrix-based characterisations of validity for all of the modal logics under consideration, and proof methods based on these characterisations which do not contain the redundancies identified in this chapter. This development is directly motivated by our treatment of similar redundancies in classical logic presented in Part I.

5.2 Notational redundancy and relevance.

This section is very brief. It is included to make the treatment of modal logic in this part of the thesis more self-contained. Basically, since the pure propositional sequent calculus forms a subsystem of all the modal logics under consideration, the redundancies identified in Part I as coming directly from the propositional structure of classical logic are also present in the search spaces generated by the modal sequent systems. We repeat certain key notions and examples. For this section we restrict our attention to the pure propositional calculus.

The rules of the pure propositional system are repeated in Figure 5–1 for the reader's convenience. To reiterate: the rules come in pairs, one pair for each connective. In each pair there is a rule for introducing the connective in the antecedent and one for introducing it in the succedent. In addition, instances of the basic sequent

$$\Gamma, A \longrightarrow A, \Delta$$

where A is an atomic formula, constitute the axioms of the system. We call the formula in which the distinguished connective is introduced, the *principal* formula of the inference. In addition, we refer to the two occurrences of the distinguished atomic formula in instances of the basic sequent as the principal formulae of that rule. The immediate subformulae of the principal formula in

$$\Gamma, A \longrightarrow A, \Delta$$

$$\frac{\Gamma \longrightarrow A, \Delta \quad \Gamma, B \longrightarrow \Delta}{\Gamma, A \Rightarrow B \longrightarrow \Delta} \Rightarrow\!\longrightarrow \qquad \frac{\Gamma, A \longrightarrow B, \Delta}{\Gamma \longrightarrow A \Rightarrow B, \Delta} \longrightarrow\!\Rightarrow$$

$$\frac{\Gamma, A, B \longrightarrow \Delta}{\Gamma, A \wedge B \longrightarrow \Delta} \wedge\!\longrightarrow \qquad \frac{\Gamma \longrightarrow A, \Delta \quad \Gamma \longrightarrow B, \Delta}{\Gamma \longrightarrow A \wedge B, \Delta} \longrightarrow\!\wedge$$

$$\frac{\Gamma, A \longrightarrow \Delta \quad \Gamma, B \longrightarrow \Delta}{\Gamma, A \vee B \longrightarrow \Delta} \vee\!\longrightarrow \qquad \frac{\Gamma \longrightarrow A, B, \Delta}{\Gamma \longrightarrow A \vee B, \Delta} \longrightarrow\!\vee$$

$$\frac{\Gamma \longrightarrow A, \Delta}{\Gamma, \neg A \longrightarrow \Delta} \neg\!\longrightarrow \qquad \frac{\Gamma, A \longrightarrow \Delta}{\Gamma \longrightarrow \neg A, \Delta} \longrightarrow\!\neg$$

Figure 5–1: A sequent calculus for pure propositional logic.

the premise(s) are called the *side* formulae of the inference. Derivations are trees of instances of the inference rules. The root of a derivation is called the *endsequent.* Proofs are derivations whose leaves are all instances of the basic sequent. A formula that occurs in a sequent of a derivation is called (after Gentzen [G69]) an *S-formula* for "sequent" formula.

Given a propositional formula A we can test its validity by attempting to construct a proof of the sequent $\longrightarrow A$ using this calculus. As usual, the best way to do this is to start with $\longrightarrow A$ and apply the rules in inverted form, *i.e.*, from conclusion to premise(s). This has the effect of constructing derivations from the root to the leaves. When inverted, the rules serve to remove occurrences of their associated connective from the sequent. The principal formula of such an (inverted) inference is said to have been *reduced.*

This defines the basic inference system. The space to be searched is that of all possible derivations of a given endsequent. We examine briefly the redundancies present in the search space generated by this simple inference system. The reader is referred to Part I for a more detailed discussion of this calculus.

The search space generated by the above inference system for the endsequent

$\longrightarrow A$ can contain many distinct interior nodes (derivations) as well as numerous proofs. Alternative paths through the space arise from the fact that when expanding a given derivation, there is a choice as to which leaf sequent to reduce next, and, for each such sequent, which of its S-formulae should be reduced.

If the formula being tested for validity is large, the intermediate derivations can themselves become very large. Moreover, there is a lot of shared structure between the derivations since formulae are repeated time and again within different sequents. This is the notational redundancy referred to above. We showed in Part I that, with the usual notion of subformula for the propositional language, every S-formula of a derivation is a subformula of the endsequent. Consequently, we can encode the interior nodes of the search space entirely in terms of the structure of the endsequent. This leads directly to the effective use of structure sharing techniques [BM72], originally developed to overcome related space problems with resolution based systems. This motivates the use of *indexed positions* in the next chapter.

In Part I we also showed that for any endsequent the search space generated has the following properties:

1. it is finite, and

2. if the endsequent is valid, every path through the space leads to a proof.

The second observation indicates that we can choose an arbitrary order of rule application and only ever consider one path through the (potentially large) search space. Taken together with the first observation we know that we can fix this order uniformly for any endsequent. In this way we can eliminate all alternative branches of the original search space leaving us with, for a given endsequent, a space containing a single path. The fact that the space is finite clearly reflects the decidability of the propositional calculus.

Although we can successfully reduce the sequent search space to a single (determinate) path, this path can be very long. Since we make an arbitrary choice of which formula to reduce next in order to expand a given derivation it is

possible to waste a lot of effort reducing formulae that cannot ever "contribute" to a proof. In Part I we defined this notion of "contribution" roughly in the following way. Every formula contains a set of atomic subformulae. An S-formula is said to *contribute* to a proof if one of its atomic subformulae appears as a principal formula of an instance of the basic sequent at a leaf of the proof.

Consider the (propositionally valid) formula: $A \wedge (B \wedge P) \Rightarrow P$, where P is an atomic formula and A and B are arbitrary formulae. The first two steps in the construction of a proof of this formula (under any uniform regime of choosing the order with which to reduce formula in the leaves of an intermediate derivation) are completely determined, leading to the derivation:

$$\dfrac{\dfrac{A,\ B \wedge P \longrightarrow P}{A \wedge (B \wedge P) \longrightarrow P}\ \wedge \longrightarrow}{\longrightarrow A \wedge (B \wedge P) \Rightarrow P}\ \longrightarrow \Rightarrow$$

At this point we have a choice: whether to reduce the formula A or the formula $B \wedge P$. Suppose our reduction regime were to always reduce the "leftmost" S-formula in the sequent. Our next step would be to reduce A. If A were some very large but valid formula we could expend a large amount of effort reducing it and its subformulae, even though it does not contribute to the proof. It should be clear that such pathological examples can be constructed whatever our reduction regime. This is the problem we have termed *relevance*, and is due to the fact that the natural search method to use with sequent calculi is based on the reduction of connectives. The method used to overcome it in the classical matrix system of Part I was the *connection*. We shall see in the next chapter that we are able to adapt this solution when the pure propositional system we have been discussing is embedded in the full modal systems.

$$\frac{\Gamma, A \longrightarrow \Delta}{\Gamma, \Box A \longrightarrow \Delta}\ \Box \longrightarrow \qquad \frac{\Gamma* \longrightarrow A, \Delta*}{\Gamma \longrightarrow \Box A, \Delta}\ \longrightarrow \Box$$

$$\frac{\Gamma*, A \longrightarrow \Delta*}{\Gamma, \Diamond A \longrightarrow \Delta}\ \Diamond \longrightarrow \qquad \frac{\Gamma \longrightarrow A, \Delta}{\Gamma \longrightarrow \Diamond A, \Delta}\ \longrightarrow \Diamond$$

$$\Gamma* \stackrel{\text{df}}{=} \{ B \mid \Box B \in \Gamma \}$$
$$\Delta* \stackrel{\text{df}}{=} \{ B \mid \Diamond B \in \Delta \}$$

Figure 5–2: Modal rules for T.

5.3 Order dependence.

As discussed in the introduction, we consider the propositional fragments of the modal sequent calculi to be formed by adding rules for manipulating modalities to the pure propositional calculus. The appropriate modal rules for the logic T are repeated in Figure 5–2 for the reader's convenience. In this section we show that the order in which these rules are used to reduce S-formulae whose major connective is a modal operator, is significant. If an inappropriate reduction order is chosen a proof may not be found. This is in direct contrast to the situation we found above for the pure propositional calculus where all paths in the search space lead eventually to a proof (if the endsequent is provable). It is however reminiscent of the addition of quantifier rules to the pure propositional calculus, as discussed in Part I.

Consider the following pair of T-derivations of the endsequent $\Box P,\ \Box(P \Rightarrow Q) \longrightarrow Q$, where we have "boxed" the principal formula of each reduction (recall that derivations are being constructed from their root to their leaves):

$$\dfrac{\dfrac{\dfrac{\boxed{Q} \longrightarrow \boxed{Q} \qquad \longrightarrow P, Q}{\boxed{P \Rightarrow Q} \longrightarrow Q}}{P, \Box(P \Rightarrow Q) \longrightarrow \boxed{\Box Q}}}{\boxed{\Box P}, \Box(P \Rightarrow Q) \longrightarrow \Box Q} \qquad \dfrac{\dfrac{P, \boxed{Q} \longrightarrow \boxed{Q} \qquad \boxed{P} \longrightarrow \boxed{P}, Q}{P, \boxed{P \Rightarrow Q} \longrightarrow Q}}{\Box P, \Box(P \Rightarrow Q) \longrightarrow \boxed{\Box Q}}$$

We cannot obtain a proof from the derivation on the left because we are unable to close one of its leaves. This problem arises because the application of the $\longrightarrow \Box$ rule restricted the formulae available for such a completion. We can influence the content of the sequent at this point by changing the order of rule application so that more (or fewer) formulae of the form $\Box A$ occur in the antecedent at the application of the $\longrightarrow \Box$ rule. This we have done in the proof on the right. The import of this observation for automated proof search is that we can no longer fix an arbitrary order for the reduction of S-formulae. The search space generated by the modal propositional calculus is therefore more complex than the one generated by the pure propositional calculus. Derivations in which the reduction order of modal formulae differ are different in an essential way. Such choices must remain explicit in the search space to retain completeness. The resulting space thus contains all possible permutations of (modal) rule applications.

The fact that reductions with the modal rules lead to formulae being "deleted" is the root cause of the order dependence we have just seen. It is somewhat different in nature from the order dependence induced by the addition of quantifier rules to the pure propositional calculus. There the problem was the introduction of distinct parameters. Here it is the maintenance of sufficient formulae in sequents to complete the proof. The effect on the structure of the search space is however similar. One of the major contributions of this thesis is the removal of this redundancy by using matrix-based techniques. The details are contained in the next chapter. We note at this point that the problem of order dependence is perhaps the most important problem to solve to achieve reasonably efficient automated proof search in modal logics. We shall see in Chapter 8 that it is precisely this problem that most of the other proof systems for modal logics, based on resolution, and suggested as appropriate for automated proof search, *fail* to solve.

$$\frac{\Gamma,\ A[a/x] \longrightarrow \Delta}{\Gamma,\ \forall x A \longrightarrow \Delta}\ \forall \longrightarrow \qquad \frac{\Gamma \longrightarrow A[a/x],\ \Delta}{\Gamma \longrightarrow \forall x A,\ \Delta}\ \longrightarrow \forall$$

$$\frac{\Gamma,\ A[a/x] \longrightarrow \Delta}{\Gamma,\ \exists x A \longrightarrow \Delta}\ \longrightarrow \exists \qquad \frac{\Gamma \longrightarrow A[a/x],\ \Delta}{\Gamma \longrightarrow \exists x A,\ \Delta}\ \exists \longrightarrow$$

For the $\longrightarrow \forall$ and $\exists \longrightarrow$ rules, the parameter a must not occur in the conclusion.

Figure 5–3: Quantifiers rules for cumulative domains.

5.4 Interactions: modal operators and quantifiers.

In the previous section we saw that the modal rules induce complex structure into the sequent search spaces. To form calculi for the quantified modal logics with cumulative domains we add the ordinary quantifier rules familiar from classical logic. The rules are repeated in Figure 5–3 for the reader's convenience. We know from Part I that the quantifier rules also induce an order dependence in the search space. In this section we investigate how these dependencies interact.

Consider the following formula of quantified modal logic:

$$\Diamond\Diamond\forall x\,(\Diamond Px \wedge \Box Qx) \Rightarrow \Diamond(\forall y Py \wedge \forall z Qz).$$

We shall attempt to demonstrate the validity of this formula using the quantified modal calculus for S4. Accordingly we modify the definitions of $\Gamma*$ and $\Delta*$ in the modal rules as follows:

$$\Gamma* \stackrel{\text{df}}{=} \{\, \Box B \mid \Box B \in \Gamma \,\}$$
$$\Delta* \stackrel{\text{df}}{=} \{\, \Diamond B \mid \Diamond B \in \Delta \,\}.$$

Starting from the endsequent:

$$\longrightarrow \Diamond\Diamond\forall x\,(\Diamond Px \wedge \Box Qx) \Rightarrow \Diamond(\forall y Py \wedge \forall z Qz)$$

there is no choice as to the first reduction, and we obtain the derivation:

$$\frac{\Diamond\Diamond\forall x\,(\Diamond Px \wedge \Box Qx) \longrightarrow \Diamond(\forall y Py \wedge \forall z Qz)}{\longrightarrow \Diamond\Diamond\forall x\,(\Diamond Px \wedge \Box Qx) \Rightarrow \Diamond(\forall y Py \wedge \forall z Qz)}$$

At this point we do have choice since both the antecedent and succedent formula of the leaf of the derivation are non-atomic. If we chose to reduce the succedent formula, the side formula of this reduction: $\forall y Py \wedge \forall z Qz$, would not survive the subsequent reduction of the antecedent formula. Looking slightly ahead, we can see that this problem will reoccur every time we reduce the $\Diamond$ operators in the antecedent formula. Since it is clear that both atomic subformulae of the succedent formula must contribute to any proof, we must preserve them. Accordingly we reduce the antecedent until we have removed all occurrences of the modal operator $\Diamond$ from it. The resulting derivation is:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{Pa,\ \Box Qa \longrightarrow \Diamond(\forall y Py \wedge \forall z Qz)}{\Diamond Pa,\ \Box Qa \longrightarrow \Diamond(\forall y Py \wedge \forall z Qz)}}{\Diamond Pa \wedge \Box Qa \longrightarrow \Diamond(\forall y Py \wedge \forall z Qz)}}{\forall x\,(\Diamond Px \wedge \Box Qx) \longrightarrow \Diamond(\forall y Py \wedge \forall z Qz)}}{\Diamond\forall x\,(\Diamond Px \wedge \Box Qx) \longrightarrow \Diamond(\forall y Py \wedge \forall z Qz)}}{\Diamond\Diamond\forall x\,(\Diamond Px \wedge \Box Qx) \longrightarrow \Diamond(\forall y Py \wedge \forall z Qz)}}{\longrightarrow \Diamond\Diamond\forall x\,(\Diamond Px \wedge \Box Qx) \Rightarrow \Diamond(\forall y Py \wedge \forall z Qz)}$$

Notice the introduction of the parameter a at the reduction of the universal quantifier $\forall x$ in the antecedent. So far so good. There are two reducible formulae in the leaf sequent of the derivation. There is no reason to prefer one over the other since the reduction of either does not affect the other formulae in the sequent. We choose to extend the derivation as follows.

$$\cfrac{\cfrac{\cfrac{\cfrac{Pa,\ Qa \longrightarrow \forall y Py \qquad Pa,\ Qa \longrightarrow \forall z Qz}{Pa,\ Qa \longrightarrow \forall y Py \wedge \forall z Qz}}{Pa,\ Qa \longrightarrow \Diamond(\forall y Py \wedge \forall z Qz)}}{Pa,\ \Box Qa \longrightarrow \Diamond(\forall y Py \wedge \forall z Qz)}}$$

At this point, we have no choice in either of the leaves; we must reduce the universal quantifiers in both succedents. In order to obtain a proof we need to introduce the parameter a in both reductions. But this violates the side condition on the quantifier rules. In order to introduce the same parameter for all the quantified variables we must reduce the quantifiers of existential force ($\forall z$

and $\forall y$) before the quantifier of universal force ($\forall x$). Going back to the sequent:

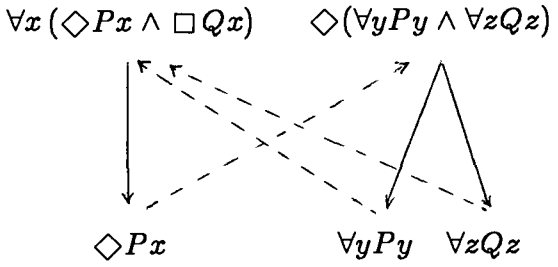
$$\forall x(\Diamond Px \wedge \Box Qx) \longrightarrow \Diamond(\forall y Py \wedge \forall z Qz)$$

we can extend the derivation as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Diamond Pa,\ \Box Qa \longrightarrow Pa}{\Diamond Pa \wedge \Box Qa \longrightarrow Pa}}{\forall x(\Diamond Px \wedge \Box Qx) \longrightarrow Pa}}{\forall x(\Diamond Px \wedge \Box Qx) \longrightarrow \forall y Py} \qquad \forall x(\Diamond Px \wedge \Box Qx) \longrightarrow \forall z Qz}}{\forall x(\Diamond Px \wedge \Box Qx) \longrightarrow \forall y Py \wedge \forall z Qz}}{\forall x(\Diamond Px \wedge \Box Qx) \longrightarrow \Diamond(\forall y Py \wedge \forall z Qz)}$$

But if we now reduce the antecedent formula $\Diamond Pa$ in the left hand leaf the succedent formula Pa will not appear in the new leaf. The same problem will clearly arise in the right hand branch if we were to perform the analogous reduction sequence there.

The problem lies in the interaction between the modal and quantifier rules. The restrictions on the modal rules force us to reduce the (antecedent) subformula $\Diamond Px$ before the (succedent) subformula $\Diamond(\forall y Py \wedge \forall z Qz)$. The restrictions on the quantifier rules force us to reduce the (succedent) subformulae $\forall y Py$ and $\forall z Qz$ before the (antecedent) formula $\forall x(\Diamond Px \wedge \Box Qx)$. But, $\Diamond Px$ is a subformula of $\forall x(\Diamond Px \wedge \Box Qx)$, and both $\forall y Py$ and $\forall z Qz$ are subformulae of $\Diamond(\forall y Py \wedge \forall z Qz)$. Consequently the modal and quantifier constraints cannot be simultaneously satisfied; *i.e.*, satisfied within the same derivation. The situation is summarised in the following diagram:



where the arrows indicate the reduction order constraints discussed above. The unsolvability of these constraints is represented by the fact that, as a directed graph, the diagram is cyclic. In fact, the original sentence:

$$\Diamond\Diamond\forall x(\Diamond Px \wedge \Box Qx) \Rightarrow \Diamond(\forall y Py \wedge \forall z Qz)$$

is not valid in the logics under consideration.

The situation for automated proof search is therefore not very healthy. The search space must necessarily contain all paths that arise from choosing different reduction orders for modal operators and quantifiers. A different order will, in general, lead to a different derivation. All such alternatives must be explored to retain completeness.

The sequent systems we are using here characterise the modal logics with cumulative domains. The effect of the cumulative domain restriction is that the same parameters may be introduced in different modal contexts as we saw in the example. The parameter a was introduced at the reduction of the (antecedent) universal quantifier $\forall x$ in our first derivation. We then went on to reduce the (antecedent) subformula $\Diamond Pa$ which notionally transfers our attention to another point in the models being investigated. (More precisely, the conclusion of the rule only fails to be forced at points where both of its antecedent formulae: $\Diamond Pa$ and $\Box Qa$, are forced, and its succedent formula: $\Diamond(\forall y Py \wedge \forall z Qz)$, fails to be forced. But, since $\Diamond Pa$ is forced at this point, by the model conditions for $\Diamond$, there must be another point, accessible from the first, at which Pa itself is forced. Pa is still taken to make sense at this new point because, since it is accessible from the point in which a was postulated to exist, and the cumulative domain condition ensures that individuals that exist at one point also exist in all points accessible from it.)

Calculi for the varying domain variants of the logics would have to encode a restriction that ensured that this form of "inheritance" did not occur. In other words the coupling between quantifiers and modalities is stronger than in the cumulative domain case that we have considered above. Parameters may only be used in the modal context in which they are introduced.

The constant domain variants on the other hand place no restrictions on the relationship between quantifiers and modal operators since the same individuals exist at every point. The only interactions occur as we have discussed above concerning the side conditions on the quantifier and modal rule applications. It seems likely (and is in fact the case) that the removal of order dependencies in

the constant domain variants will be easier than in the other variants, since the coupling of quantifiers with modalities is weaker.

5.5 Conclusions.

In this brief chapter we have investigated the structure of the search spaces generated by the modal sequent calculi presented in Chapter 4. We have demonstrated that these search spaces contain a number of familiar redundancies, namely:

1. notational redundancies,

2. relevance: the presence of inessential choices, and

3. order dependence: the sensitivity of the search space to the order of application of certain inference rules.

We expected the first two types of redundancy since it was shown in Part I that these redundancies are due to the basic sequent format of the pure propositional calculus. This calculus forms a subsystem of the modal calculi.

The sensitivity of the search space to rule application order was shown to be more complex than classical logic. Both the modal rules and the quantifier rules introduce order dependencies. We showed that these dependencies interact in a major way to complicate the search space.

In the next chapter, based on these observations and the methods used in Part I to overcome such redundancies, we present matrix based characterisations of validity for modal logics, and proof methods based on these characterisations. The search spaces generated by the matrix-based proof systems do not contain the redundancies identified above.

Since we only have at our disposal sequent calculi for the cumulative domain variants of some of the modal logics we are interested in, it might seem as though we will be unable to develop efficient proof methods for the other variants and

logics by means of the analysis we have undertaken. In fact, the matrix-based proof methods we develop extend to all the logics and variants under consideration. We discuss the implications of this surprising fact further in Chapter 10.

Chapter 6

Matrix characterisations of validity in modal logics.

6.1 Introduction.

In Chapter 4 we presented standard cut-free sequent calculi for the cumulative domain variants of the first-order modal logics K, K4, D, D4, T, S4. In the previous chapter we investigated the combinatorial properties of modalities by studying the search spaces induced by these calculi. In this chapter, based on this proof-theoretic analysis and the ideas developed for classical logic reported in the first part of the thesis, we develop matrix-based characterisations of validity for the modal logics K, K4, D, D4, T, S4 and S5. Our methods are sufficiently general to capture the standard first-order versions of the logics (varying domains), as well as both constant and cumulative domain variants, and thus transcend the scope of the cut-free sequent systems that led to them. We prove the correctness and completeness of the matrix characterisations. In the next chapter we outline efficient proof systems based on the matrix characterisations, and discuss their use as decision procedures for the propositional fragments of the logics.

We emphasise once more that our concern is the efficient automation of the modal logics rather than detailed arguments for their use in a particular application. Such arguments can be found in the literature cited. In Chapter 8

we compare the matrix proof systems with other proof systems for modal logics proposed in the literature as suitable for automated proof search. We show that the matrix systems have considerable computational advantages over these other (typically resolution based) systems.

The remainder of this introduction is devoted to an overview of the matrix-based characterisations developed below. It serves to outline the structure of this chapter.

6.1.1 Overview.

The modal sequent calculi presented in Chapter 4 and analysed in Chapter 5 bear a close relationship to the classical sequent calculus discussed in the first part of this thesis. Indeed, the modal systems are formed by adding to the classical calculus rules specifically for manipulating modalities. As we have seen, this seemingly innocuous extension has far reaching effects on sequent-based proof search. Let us concentrate first on the features of the classical system that are preserved by the addition of the modal rules, namely:

- the systems are still cut-free,
- they retain the subformula property, and
- the basic sequent is unchanged.

The presence of the first two features in the classical sequent calculus were shown to enable the encoding of derivations using syntactic references or pointers to the endsequent. Such syntactic references were called *positions*. This encoding supported the use of particular implementation techniques such as structure-sharing [BM72]. The fact that these features are also present in the modal sequent calculi indicates that we can (and will) develop similar techniques for the modal logics. This motivates the definition of *formula trees*, *positions* and *polarity* presented in §6.2.1 below.

We noted in the previous chapter that modal operators of necessary force are *generative* in the same sense as the first-order quantifiers analysed in Part I. Thus, a positive occurrence of the generative subformula $\Box A$ in the endsequent may give rise to multiple instances of its immediate subformula A in a derivation. We adapt Bibel's method of encoding this sort of duplication for quantifiers [Bib82a]. The key notion is that of a *modal multiplicity*. The details are discussed in §6.2.2. The form of this solution supports the implementation of the resulting matrix systems using structure-sharing techniques. It also ensures that such duplication can be demand-driven, rather than adhoc, as we show in the next chapter.

We have demonstrated that the combinatorics of the inference rules dealing with the propositional structure of classical logic can be captured by the matrix representation of formulae, due originally to Prawitz [Pra60] (for formulae in conjunctive normal form) and generalised by Andrews [And81], and Bibel [Bib81]. Since the modal logics share the propositional structure of classical logic, the notion of *path* is generalised simply from the classical to the modal case with the modal operators acting in a similar way to quantifiers with respect to this structure. The details are presented in §6.2.3 below.

Since the classical basic sequent is preserved in the modal systems, we infer that validity in the modal logics can be characterised, at least partially, in terms of a set of *connections* that *span* the formulae viewed as a matrix. (That is to say: every path through the formula contains a connection from the set.) So far so good. But now we must deal with the conditions under which such a pair of atomic formulae of opposite polarity within a formula can be deemed to be *complementary*, *i.e.*, not only deemed to correspond to an instance of the basic sequent, but also that a correct derivation can be formed with the endsequent at its root, and that instance of the basic sequent at one of its leaves.

A simple example will help at this point. Consider the formula: $\Box P \Rightarrow \Box\Box P$, where P is some arbitrary (atomic) formula. We showed in Chapter 4 that such formulae are valid in S4, but not in K. For the reader's convenience the S4-proof and a K-derivation are repeated in Figure 6–1. Treating the modal

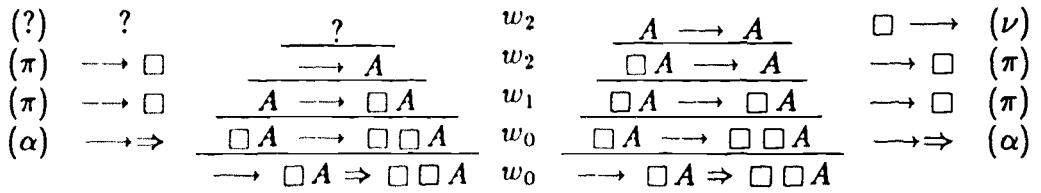


Figure 6–1: K-derivation (left) and S4-proof (right) of $\Box P \Rightarrow \Box\Box P$.

operators as structurally equivalent to quantifiers for the moment (*i.e.*, the paths through $\Box A$ and $\Diamond A$ are simply the paths through A itself) the formula contains only one path and the two occurrences of P occur on that path with opposite polarity. The occurrences therefore form a connection. But here we see that in one modal logic (S4) we may consider the connection to be complementary, but in another (K) we must not. We must define a notion of complementarity for atomic formula occurrences, *for each logic*, such that, given a spanning set of complementary connections, we are ensured of the existence of a proof of the formula in the appropriate sequent calculus for that logic. Roughly speaking, the set of connections form the leaves of this proof.

Recall that in the case of classical logic the premise and conclusion of a sequent rule could be viewed as making assertions about the structure of a given model. Recall also that in Chapter 4 we indicated that the same is true for the modal logics but that a modal model consists of a set of classical models or points. It is therefore possible (and usual) for a proposition to be forced at one point and fail to be forced at another, with no contradiction arising. For the π type inference rules, $\longrightarrow \Box$ and $\Diamond \longrightarrow$, the premise refers to a different point than the conclusion, the former being accessible from the latter. For instance, viewed from the root upwards, each (inverted) application of a π-rule in our example can be said to introduce a new point accessible from (at least) the point associated with the conclusion. Now, let us name these points explicitly for

our example. Call the point associated with the endsequent w_0. There are two applications of (inverted) π-rules, which introduce two new points w_1 and w_2 respectively. We have that $w_0 \; R \; w_1$ and $w_1 \; R \; w_2$.

Different logics allow us to conclude different things about the new points introduced in this manner. Recall the model condition for the modal operator $\Box$:

- $w \Vdash \Box A$ iff for all v such that $w \; R \; v$, $v \Vdash A$, or equivalently,
- $w \nVdash \Box A$ iff for some v such that $w \; R \; v$, $v \nVdash A$.

In K then, all we can infer is that if $w_0 \Vdash \Box P$, $w_1 \Vdash P$ and hence the subderivation:

$$\frac{P \longrightarrow \Box P \qquad w_1}{\Box P \longrightarrow \Box\Box P \qquad w_0}$$

of the K-derivation. Likewise, since there is no antecedent formula $\Box A$ (reflecting the fact that we cannot assert that $w_1 \Vdash \Box A$), the subsequent application of the inverted π-rule results in:

$$\frac{\longrightarrow P \qquad w_2}{P \longrightarrow \Box P \qquad w_1}$$

For S4 however, since the accessibility relation is transitive, we have that $w_0 \; R \; w_2$. Consequently if $w_0 \Vdash \Box P$ then $v \Vdash \Box P$ whenever $w_0 \; R \; v$. We can therefore "preserve" the antecedent formula $\Box P$ through the applications of the inverted π-rule in the S4-derivation thus:

$$\frac{\Box P \longrightarrow \Box P \qquad w_1}{\Box P \longrightarrow \Box\Box P \qquad w_0} \quad \text{and} \quad \frac{\Box P \longrightarrow P \qquad w_2}{\Box P \longrightarrow \Box P \qquad w_1}$$

Again, since in S4 the accessibility relation is reflexive, $w_2 \Vdash \Box P$ implies $w_2 \Vdash P$. This inference is performed by the (inverted) ν-rule, and the S4-proof is complete.

The key observation is that an appropriate notion of complementarity can be defined by noting the context of atoms relative to the modal operators in the endsequent. We will represent these contexts by *prefixes*, and will consider

a connection to be complementary when there is a mapping on the prefixes of the atomic formulae of the connection which renders them identical. These mappings can in fact be seen as *substitutions* in the usual sense. The logic-dependence of such a notion of complementarity is reflected in the conditions such mappings should satisfy. For a logic $\mathcal{L}$, these conditions define the set of $\mathcal{L}$-*admissible* substitutions. In practice, as discussed in the next chapter, $\mathcal{L}$-admissible substitutions may be computed by specialised unification algorithms, one for each logic.

In the example, the first occurrence of the atom P in the antecedent of the implication of the endsequent, occurs inside one modal operator. The polarity of this operator is such that only ν-rules may be applied to it. We give the atom a *prefix* of $\bar{a}$. The other occurrence of P in the endsequent occurs within two modalities, both of which can only have π-rules applied to them. We therefore give it a prefix of the form bc: a sequence (or string) comprising the two atomic prefixes b and c. We treat $\bar{a}$ as a variable wih respect to substitutions, and b and c as constants. (Notice that, by convention here and in the sequel, we distinguish variables from constants by means of an overbar.) Roughly speaking, the atomic formulae occurrences forming the connection are deemed to be K-complementary if the prefixes of the atoms are unifiable under the restriction that variables may only be mapped to individual constants. The unification problem $\langle \bar{a}, bc \rangle$ is clearly not solvable under this restriction. Roughly speaking, the atomic formulae occurrences forming the connection are deemed to be S4-complementary if their prefixes are unifiable under an equational theory in which variables may be instantiated to entire sequences. The unification problem $\langle \bar{a}, bc \rangle$ is solvable in this theory, with unifier $\bar{a} \mapsto bc$. The details are presented in §6.2.4.

Lifting these results to first-order constant domain modal logics is simply a matter of combining the modal notion of complementarity with the first-order notion presented in Part I. For the varying domain variants, we index individual variables with the prefix of their quantifier. Roughly speaking, a given parameter may be substituted for two individual variables just in case the prefixes of their respective quantifiers unify under the modal substitution referred to above. For

the cumulative domain variants, a similar restriction is imposed except that we require the quantifier prefix of universally quantified variables in such pairs be accessible from the quantifier prefix of its existentially quantified partner. In both cases the condition depends on the properties of the accessibility relation of the particular logic. The details are given in §6.2.4.

Sections 6.3 and 6.4 are devoted to proving the correctness and completeness, respectively, of the matrix-based characterisations of validity for all logics and variants under consideration. The proofs follow the pattern established in Part I; *i.e.*, they are based on correctness and (systematic) completeness proofs for sequent calculi and analytic tableau proof systems [Smu68,Kle68,Fit83].

Testing a formula for validity within a modal logic is therefore successfully reduced to a process of path checking and complementarity tests as in the classical case. This means that search strategies developed for the classical matrix system are applicable to the modal matrix systems *without alteration*. There are two major differences however:

- The test for the complementarity of a connection is performed by a specialised unification algorithm. For some logics the appropriate unification problems do not yield a single most general unifier. However, it turns out that, for the logics under consideration here, the set of most general unifiers is finite.

- The generative modal operators give rise to extra opportunities for duplication during proof search. This can be seen as the basic cause for the increased complexity (PSPACE) of the decision problem for the propositional fragments of the modal logics K, K4, D, D4, T, S4 over the propositional fragment of classical logic (NP). S5, as usual, is a special case (NP).

In Chapter 7 we discuss search strategies of particular use for individual modal systems and present examples. We pay particular attention to the structure of the unification algorithms necessary to support the complementarity tests. As an additional topic we outline how efficient decision procedures for the propo-

sitional fragments of the modal logics can be developed based on the matrix characterisations.

We now present the details.

6.2 Matrices, paths and connections.

In this section we develop matrix-based characterisations of validity for the modal logics under consideration. The proofs of the correctness and completeness of these characterisations can be found in §6.3 and §6.4 respectively. This section is written so as to be technically self-contained. On the other hand, we have relegated motivational information to short remarks, since the introduction above and the detail in Part I should suffice.

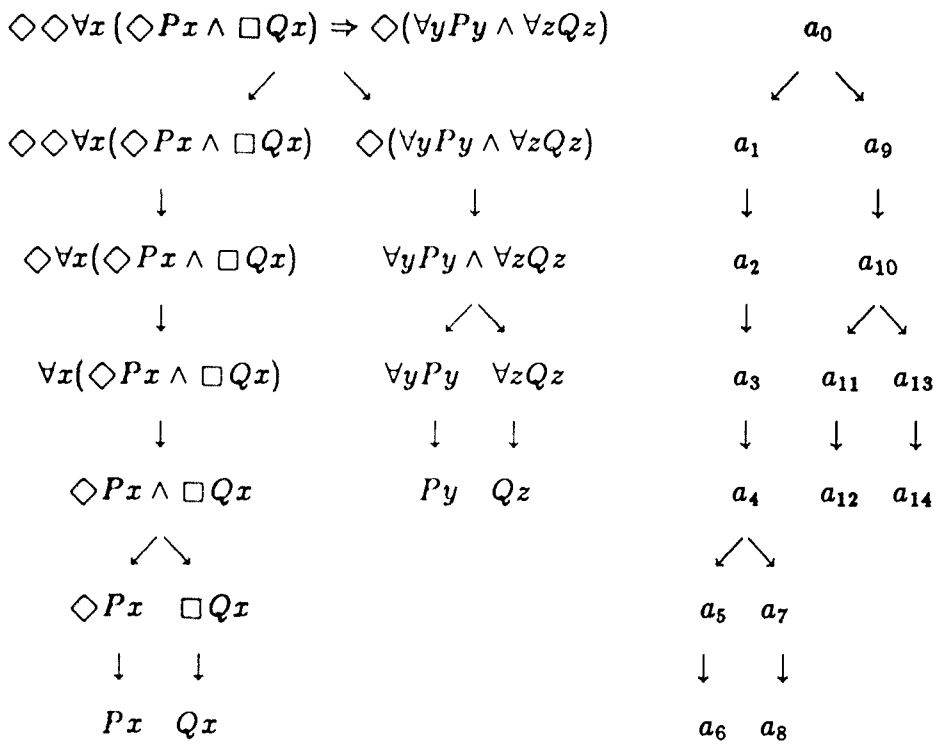
6.2.1 Formula occurrences.

The formation tree for a formula is, as usual, a tree indicating how the formula is built up from its subformulae. A *formula tree* for a signed formula $X = \langle A, n\rangle$ is a tree of names, or *positions*, one for each distinct subformula occurrence in the formation tree of A. The formation tree and a formula tree for the signed formula:

$$\langle \Diamond\Diamond\forall x\,(\Diamond Px \wedge \Box Qx) \Rightarrow \Diamond(\forall y Py \wedge \forall z Qz), 0\rangle,$$

are shown in Figure 6–2.

We use k and l, possibly subscripted, as metavariables to range over the positions of formula trees. With each position, k, of such a formula tree we associate a *label*, denoted $\mathrm{lab}(k)$, and a *polarity*, denoted $\mathrm{pol}(k)$. The label of a position is the formula occurrence appearing at the corresponding point in the formation tree for the formula. The polarity of a position of a formula tree for $\langle A, n\rangle$ is n, if its label occurs positively in A, and $(n + 1) \bmod 1$, if its label occurs negatively in A. The labels and polarities of the positions of the example formula tree are also shown in Figure 6–2.



k	pol(k)	lab(k)	Ptype(k)	Stype(k)
a_0	0	$\Diamond\Diamond\forall x(\Diamond Px \wedge \Box Qx) \Rightarrow \Diamond(\forall yPy \wedge \forall zQz)$	α	π_0
a_1	1	$\Diamond\Diamond\forall x(\Diamond Px \wedge \Box Qx)$	π	α_1
a_2	1	$\Diamond\forall x(\Diamond Px \wedge \Box Qx)$	π	π_0
a_3	1	$\forall x(\Diamond Px \wedge \Box Qx)$	γ	π_0
a_4	1	$\Diamond Px \wedge \Box Qx$	α	γ_0
a_5	1	$\Diamond Px$	π	α_1
a_6	1	Px	–	π_0
a_7	1	$\Box Qx$	ν	α_2
a_8	1	Qx	–	ν_0
a_9	0	$\Diamond(\forall yPy \wedge \forall zQz)$	ν	α_2
a_{10}	0	$\forall yPy \wedge \forall zQz$	β	ν_0
a_{11}	0	$\forall yPy$	δ	β_1
a_{12}	0	Py	–	δ_0
a_{13}	0	$\forall zQz$	δ	β_2
a_{14}	0	Qz	–	δ_0

Figure 6–2: Example formation and formula tree.

Positions with atomic labels are called *atomic positions.* We use $\ll$ to denote the tree ordering of positions in the formula tree: *i.e.*, for positions k and l, $k \ll l$ just in case $\mathrm{lab}(l)$ is a subformula of $\mathrm{lab}(k)$ in the formation tree.

REMARKS. Positions form the basis for implementations of the matrix systems using structure-sharing techniques [BM72]. A position should be interpreted as a pointer to a subformula of the formula being tested for validity stored in computer memory.

Theoretically, we use an extension of the formula tree for $\langle A, n\rangle$ to represent the possible formula that may take part in a sequent derivation of an endsequent of the form: $\longrightarrow A$ if $n = 0$, and $A \longrightarrow$ if $n = 1$. In this spirit, the polarity of a position determines whether the subformula labelling the position will occur as an antecedent or succedent formula of any such derivation. (END OF REMARKS.)

Each position k represents a particular signed formula, denoted $\mathrm{sform}(k)$, as follows:

$$\mathrm{sform}(k) = \langle \mathrm{lab}(k)\,, \mathrm{pol}(k)\rangle.$$

Consequently, the classification of signed formulae presented in Chapter 4 can be extended to the non-atomic positions of a formula tree by defining the ***principal type***, $\mathrm{Ptype}(k)$, of a position k to be the type of the signed formula $\mathrm{sform}(k)$. The principal types are therefore α, β, γ, δ, ν and π. (Note: atomic positions have no principal type.)

Every formula occurrence of the formation tree, except the root formula itself, is an immediate subformula of some subformula of the root formula. Consequently we can define the *secondary type*, $\mathrm{Stype}(k)$, of a position k to be the type of the signed formula $\mathrm{sform}(k)$ determined with respect to the principal type of its parent. For example: suppose $\mathrm{sform}(k) = \langle B \wedge C, 0\rangle$. Suppose also that k_1 and k_2 are the children of k in the formula tree, *i.e.*,

$$\mathrm{sform}(k_1) = \langle B, 1\rangle \quad \text{and} \quad \mathrm{sform}(k_2) = \langle C, 0\rangle.$$

Then, since $\mathrm{Ptype}(k) = \alpha$ we define

$$\mathrm{Stype}(k_1) = \alpha_1 \quad \text{and} \quad \mathrm{Stype}(k_2) = \alpha_2.$$

Another example: suppose $\mathrm{sform}(k) = \langle \Diamond B, 1 \rangle$. Suppose also that k_1 is the child of k with $\mathrm{sform}(k_1) = \langle B, 1 \rangle$. Then, since $\mathrm{Ptype}(k) = \pi$, we have $\mathrm{Stype}(k_1) = \pi_0$.

The secondary types are therefore α_1, α_2, β_1, β_2, γ_0, δ_0, ν_0 and π_0. In addition, if the polarity of the root position is 0 we define its secondary type to be π_0. Conversely, if the polarity of the root position is 1 we define its secondary type to be ν_0.

Each non-atomic position therefore has two types:

- its principal type (*eg.*, $\alpha, \beta, \nu, \ldots$): determined by its label and polarity, and
- its secondary type (*eg.*, $\alpha_1, \alpha_2, \beta_1, \ldots$): determined by the principal type of its parent.

Atomic positions have only a secondary type. The principal and secondary types of the positions of the example formula tree are also shown in Figure 6–2.

For a given formula tree, we shall use $\mathcal{V}_0$, Π_0, Γ_0 and Δ_0 to denote the sets of positions of secondary type ν_0, π_0, γ_0 and δ_0 respectively.

6.2.2 Multiplicities.

We noted above that our intention is to use formula trees to represent sequent derivations. We shall characterise the existence of a sequent proof of the formula, and hence its validity, in terms of the internal structure of the formula, as we did in Part I for classical logic. Positions are the means by which we refer to this structure. The notions of *multiplicities* and *indexed* formula trees introduced below, are used to capture the fact that during a derivation certain subformulae are utilised in multiple ways. We say that quantifiers of universal force, and modal operators of necessary force, are *generative*. In terms of the sequent calculi of Chapter 4 the genericity of such quantifiers and modal operators emerges through the γ and ν rules which, when inverted, give rise to multiple instances of their side formulae.

In Part I we presented a notion of multiplicity for the quantifiers. We repeat that here in the context of modal logic, but call it a *first-order* multiplicity. We treat the generative modal operators in a similar manner leading to the notion of a *modal* multiplicity. A multiplicity for a formula of quantified modal logic is simply the combination of both the first-order and modal multiplicities.

Informally, a multiplicity indicates how many instances of particular subformulae are, or may be, utilised in a derivation. Recall that a position, k, of the basic formula tree represents a (sub)formula via its label: $\mathrm{lab}(k)$. We distinguish different instances of this formula by indexing the position thus: k^κ, where κ is sequence of positive integers. We arrange that $\mathrm{lab}(k^\kappa)$ represents a distinct instance of $\mathrm{lab}(k)$ for each distinct index κ. That is:

$$\mathrm{lab}(k^\kappa) = \mathrm{lab}(k^\tau) \quad \text{iff} \quad \kappa = \tau.$$

Multiplicities are the means by which we generate appropriate indices.

For quantified formulae such as $\mathrm{lab}(k) = \forall x B$, instances of its immediate subformula: $\mathrm{lab}(k_1) = B$, are formed by the substitution of distinct constants or parameters for the individual variable x free in B. While the positions allow us to name these instances, we also require the labels of indexed positions to represent the different subformulae as mentioned above. Consequently, we define the label of an indexed position to take account of these potential substitutions. In this we are aided by our particular formulation of multiplicity (as opposed to Bibel's) as argued in Part I. We use the γ_0 and δ_0 positions themselves as an indication of the potential for substitution to form a new instance.

The following definitions are introduced for a given formula tree for a given signed formula $X = \langle A, 0 \rangle$.

A function μ_M from $\mathcal{V}_0$ to the natural numbers is called a *modal multiplicity* for X; it serves to encode the number of instances of subformulae of X in the scope of a modal operator of necessary force considered within a putative proof.

A function μ_Q from Γ_0 to the natural numbers is called a *first-order multiplicity* for X; it serves to encode the number of instances of subformulae of X in the scope of a quantifier of universal force considered within a putative proof.

A *multiplicity* μ for X is the combination of a modal and first-order multiplicity thus: for a position $k \in \mathcal{V}_0 \cup \Gamma_0$:

$$\mu(k) = \begin{cases} \mu_M(k), & k \in \mathcal{V}_0; \\ \mu_Q(k), & k \in \Gamma_0. \end{cases}$$

If μ is a multiplicity for X we define the (indexed) formula tree for the *indexed formula* X^μ as a tree of indexed positions of the form k^κ, where k is a position of the basic formula tree for X and κ is a sequence of positive integers defined in the manner described below. Let $k_1 \ll k_2 \ll \cdots \ll k_n \trianglelefteq k$, $1 \leq n$, be all those elements of Γ_0 that dominate k in the formula tree for X. The indexed position k^κ is an position of the indexed formula tree for X^μ provided:

1. k is a position of the formula tree for X.

2. $\mu(k_i) \neq 0$, $1 \leq i \leq n$.

3. $\kappa = m_1 m_2 \cdots m_n$ where $1 \leq m_i \leq \mu(k_i)$, $1 \leq i \leq n$.

We shall use $\kappa \prec \tau$ to denote that κ is a proper initial sequence of τ. The ordering on the underlying tree is extended to the indexed tree as follows: for indexed positions k^κ and l^τ,

$$k^\kappa \ll^\mu l^\tau \qquad \text{iff} \qquad k \ll l \quad \text{and} \quad \kappa \preceq \tau$$

i.e., k must dominate l in the (unindexed) formula tree, and κ must be an initial (possibly not proper) sequence of τ. The polarity, $\mathrm{pol}(k^\kappa)$, of an indexed position k^κ is taken to be the same as the polarity of its underlying (unindexed) position k; *i.e.*, $\mathrm{pol}(k^\kappa) = \mathrm{pol}(k)$. The label, $\mathrm{lab}(k^\kappa)$, of an indexed position k^κ is defined inductively as follows:

1. $\mathrm{lab}(k_0) = A$, if k_0 is the root position of the formula tree.

2. If $\mathrm{lab}(k^\kappa) = B \wedge C$, and k_1^κ, k_2^κ are the children of k^κ, then $\mathrm{lab}(k_1^\kappa) = B$ and $\mathrm{lab}(k_2^\kappa) = C$.

3. If $\mathrm{lab}(k^\kappa) = B \vee C$, and k_1^κ, k_2^κ are the children of k^κ, then $\mathrm{lab}(k_1^\kappa) = B$ and $\mathrm{lab}(k_2^\kappa) = C$.

4. If $\mathrm{lab}(k^\kappa) = B \Rightarrow C$, and k_1^κ, k_2^κ are the children of k^κ, then $\mathrm{lab}(k_1^\kappa) = B$ and $\mathrm{lab}(k_2^\kappa) = C$.

5. If $\mathrm{lab}(k^\kappa) = \neg B$, and k_1^κ is the child of k^κ, then $\mathrm{lab}(k_1^\kappa) = B$.

6. If $\mathrm{lab}(k^\kappa) = \forall x B$, and k_1^τ is a child of k^κ, for some τ, $\kappa \preceq \tau$, then $\mathrm{lab}(k_1^\tau) = B[k_1^\tau/x]$.

7. If $\mathrm{lab}(k^\kappa) = \exists x B$, and k_1^τ is a child of k^κ, for some τ, $\kappa \preceq \tau$, then $\mathrm{lab}(k_1^\tau) = B[k_1^\tau/x]$.

8. If $\mathrm{lab}(k^\kappa) = \Diamond B$, and k_1^τ is a child of k^κ, for some τ, $\kappa \preceq \tau$, then $\mathrm{lab}(k_1^\tau) = B$.

9. If $\mathrm{lab}(k^\kappa) = \Box B$, and k_1^τ is a child of k^κ, for some τ, $\kappa \preceq \tau$, then $\mathrm{lab}(k_1^\tau) = B$.

That is, we use the position itself as a marker for where substitutions can be performed for individual variables. Positions of γ_0 and δ_0-type appear in the atoms labelling the atomic positions of the indexed formula tree in place of what otherwise would be free individual variables.

The notation $\mathrm{sform}(u)$ is extended to indexed positions in the obvious way, namely:

$$\mathrm{sform}(u) = \langle \mathrm{lab}(u), \mathrm{pol}(u) \rangle.$$

Consequently, since the polarity of an indexed position k^κ, and the structural form of its label, is identical to the polarity and form of the label of the underlying position k, k^κ inherits the types (both principal and secondary) of k. We use $\mathcal{V}_0(\mu)$, $\Pi_0(\mu)$, $\Gamma_0(\mu)$ and $\Delta_0(\mu)$ to denote the sets of indexed positions of secondary type ν_0, π_0, γ_0 and δ_0 respectively, in an indexed formula tree for X^μ.

Figure 6–3 shows an indexed formula tree for the example formula of Figure 6–2 with a multiplicity of $\mu_Q(a_4) = 2$ and 1 otherwise. As a convention we

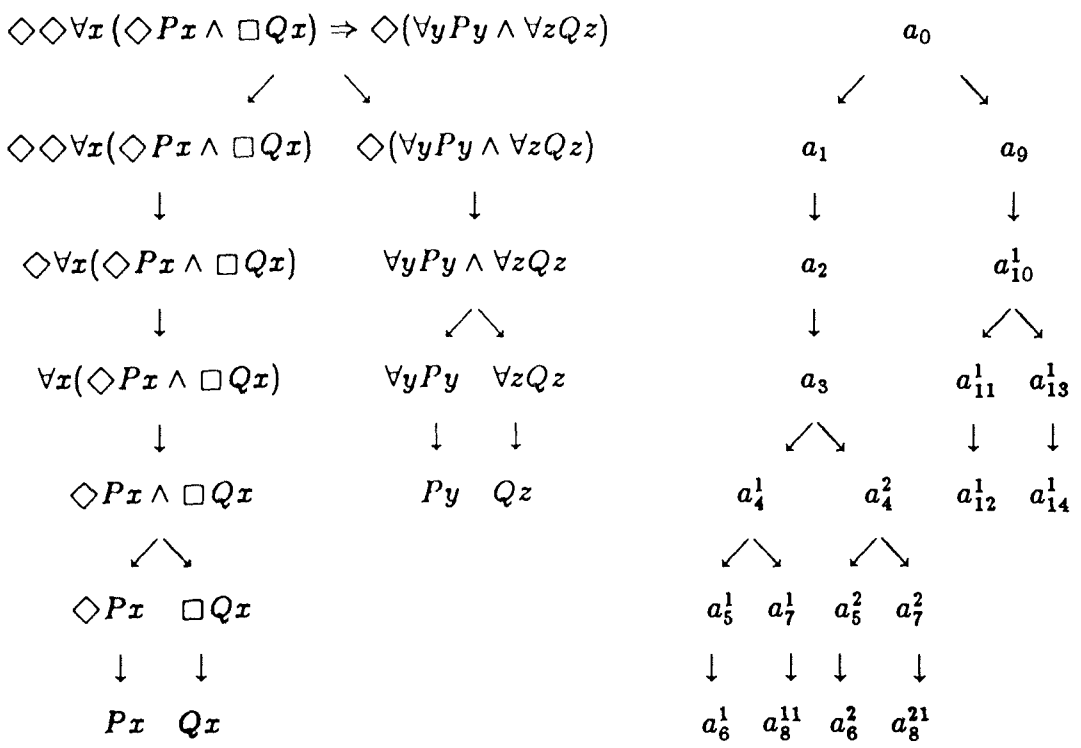


Figure 6–3: Example indexed formula tree.

omit indices consisting of the empty sequence and omit the brackets surrounding sequences of integers. Hence 12 denotes the sequence consisting of the unit sequences 1 and 2. Since we will have no need to consider multiplicities higher than nine in the development of the matrix systems, this should not lead to any confusion. Figure 6–4 shows the polarities, labels and types of the tree.

We shall use u and v, possibly subscripted, as meta-variables ranging over indexed positions when we are not interested in the index, and drop the superscript on $\ll^{\mu}$. Moreover we use $\alpha, \alpha_1, \alpha_2, \beta, \ldots$, *etc*, as before to denote arbitrary indexed positions of that (principal or secondary) type. We shall feel free to use such notation as α^{κ} to denote an indexed position of α-type when we wish to identify the index. Henceforth, we shall refer to indexed positions simply as positions.

REMARKS. In Part I we slightly altered Bibel's notion of a multiplicity

u	pol(u)	lab(u)	Ptype(u)	Stype(u)
a_0	0	$\Diamond\Diamond\forall x\,(\Diamond Px \wedge \Box Qx) \Rightarrow \Diamond(\forall y Py \wedge \forall z Qz)$	α	π_0
a_1	1	$\Diamond\Diamond\forall x(\Diamond Px \wedge \Box Qx)$	π	α_1
a_2	1	$\Diamond\forall x(\Diamond Px \wedge \Box Qx)$	π	π_0
a_3	1	$\forall x(\Diamond Px \wedge \Box Qx)$	γ	π_0
a_4^1	1	$\Diamond Pa_4^1 \wedge \Box Qa_4^1$	α	γ_0
a_5^1	1	$\Diamond Pa_4^1$	π	α_1
a_6^1	1	Pa_4^1	–	π_0
a_7^1	1	$\Box Qx$	ν	α_2
a_8^{11}	1	Qa_4^1	–	ν_0
a_4^2	1	$\Diamond Pa_4^2 \wedge \Box Qa_4^2$	α	γ_0
a_5^2	1	$\Diamond Pa_4^2$	π	α_1
a_6^2	1	Pa_4^2	–	π_0
a_7^2	1	$\Box Qa_4^2$	ν	α_2
a_8^{21}	1	Qa_4^2	–	ν_0
a_9	0	$\Diamond(\forall y Py \wedge \forall z Qz)$	ν	α_2
a_{10}^1	0	$\forall y Py \wedge \forall z Qz$	β	ν_0
a_{11}^1	0	$\forall y Py$	δ	β_1
a_{12}^1	0	Pa_{12}^1	–	δ_0
a_{13}^1	0	$\forall z Qz$	δ	β_2
a_{14}^1	0	Qa_{14}^1	–	δ_0

Figure 6–4: Polarities, labels and types for an indexed formula tree.

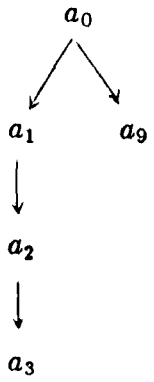


Figure 6–5: Indexed formula tree with constant zero multiplicity.

[Bib82a]. The resulting notion corresponds to a first-order multiplicity here. The new definition is more appropriate for discussing the rôle of quantifiers in the matrix and sequent systems presented there. Here we have another vindication of our formulation since we are able to treat the genericity of modalities and quantifiers (technically) in exactly the same way.

We have left open the possibility that the multiplicity of a position may be zero. In this case the indexed formula tree is truncated. Figure 6–5 shows the indexed formula tree of our example with a constant zero multiplicity. (END OF REMARKS.)

6.2.3 Paths and connections.

IMPORTANT NOTATIONAL POINT. We warn the reader that we shall systematically abuse our notation and use the names of types to denote arbitrary (un)indexed positions of that type within formal definitions such as the definition of the notion of path below. In particular, if we say: "if s, π^κ is a path..." we mean that "if s, u is a path, and $\mathrm{Ptype}(u) = \pi$,..." Furthermore, in this context we shall use π_0^κ to denote the child of u. Similar abuses are extended to the other types. We shall include indices explicitly where necessary. (END OF POINT.)

Let X^μ be an indexed formula. A *path* through X^μ is a subset of the positions of its formula tree defined below. We shall use s and t, possibly subscripted, to denote paths, and adopt the notation s, u to denote the path (set) $s \cup \{u\}$. The set of paths through X^μ, is the smallest set such that:

1. $\{ k_0 \}$ is a path, where k_0 is the root position of the formula tree for X^μ;

2. if s, α^κ is a path, so is $(s \setminus \{\alpha^\kappa\}), \alpha_1{}^\kappa, \alpha_2{}^\kappa$;

3. if s, β^κ is a path, so are $(s \setminus \{\beta^\kappa\}), \beta_1{}^\kappa$ and $(s \setminus \{\beta^\kappa\}), \beta_2{}^\kappa$;

4. if s, γ^κ is a path, so is $s, \gamma_0{}^{\kappa j}$, for any j, $1 \leq j \leq \mu_Q(\gamma_0)$;

5. if s, δ^κ is a path, so is $(s \setminus \{\delta^\kappa\}), \delta_0{}^\kappa$.

6. if s, ν^κ is a path, so is $s, \nu_0{}^{\kappa j}$, for any j, $1 \leq j \leq \mu_M(\nu_0)$;

7. if s, π^κ is a path, so is $(s \setminus \{\pi\}), \pi_0{}^\kappa$.

The path:

$$(s \setminus \{\alpha^\kappa\}), \alpha_1{}^\kappa, \alpha_2{}^\kappa$$

is said to have been *obtained by reduction on* α^κ from s, α^κ. Similarly in the other cases. Notice that in the generative cases: γ and ν, there is a choice as to which child of the position to introduce. The children differ solely in the last element of their indices.

EXAMPLE. Consider the indexed formula tree of Figure 6–3. A sample of the paths through this indexed formula are shown in Figure 6–6. Apart from the first path, the rest are obtained by a single reduction from their predecessor. Notice how the generative positions (*i.e.*, those of γ and ν type) are preserved once they enter the path. Notice also how their indexed children are introduced by repeated reduction. Finally, notice how the β-red on a^1_{10} introduces only one of the children of that position, namely: a^1_{11}. Other paths result from the inclusion of the alternative child: a^1_{13}. The resemblence of the reduction operations used

$$\{\, a_0 \,\}$$

α-reduction on a_0

$$\{\, a_1, a_9 \,\}$$

ν-reduction on a_9

$$\{\, a_1, a_9, a_{10}^1 \,\}$$

β-reduction on a_{10}^1

$$\{\, a_1, a_9, a_{11}^1 \,\}$$

δ-reduction on a_{11}^1

$$\{\, a_1, a_9, a_{12}^1 \,\}$$

π-reduction on a_1

$$\{\, a_2, a_9, a_{12}^1 \,\}$$

π-reduction on a_2

$$\{\, a_3, a_9, a_{12}^1 \,\}$$

γ-reduction on a_3

$$\{\, a_3, a_4^1, a_9, a_{12}^1 \,\}$$

α-reduction on a_4^1

$$\{\, a_3, a_5^1, a_7^1, a_9, a_{12}^1 \,\}$$

π-reduction on a_5^1

$$\{\, a_3, a_6^1, a_7^1, a_9, a_{12}^1 \,\}$$

ν-reduction on a_7^1

$$\{\, a_3, a_6^1, a_7^1, a_8^{11}, a_9, a_{12}^1 \,\}$$

γ-reduction on a_3

$$\{\, a_3, a_4^2, a_6^1, a_7^1, a_8^{11}, a_9, a_{12}^1 \,\}$$

α-reduction on a_4^2

$$\{\, a_3, a_5^2, a_7^2, a_6^1, a_7^1, a_8^{11}, a_9, a_{12}^1 \,\}$$

π-reduction on a_5^2

$$\{\, a_3, a_6^2, a_7^2, a_6^1, a_7^1, a_8^{11}, a_9, a_{12}^1 \,\}$$

ν-reduction on a_7^2

$$\{\, a_3, a_6^2, a_7^2, a_8^{21}, a_6^1, a_7^1, a_8^{11}, a_9, a_{12}^1 \,\}$$

Figure 6–6: Paths through the indexed formula of Figure 6–3.

to define the notion of path to the sequent rules of Chapter 4 is not accidental. (END OF EXAMPLE.)

Each path s through X determines a set of positions as follows:

$$S(s) \stackrel{\text{df}}{=} \{ v \mid v \preceq u \quad \text{for some } u \in s \}.$$

We call $S(s)$ the set *associated with* the path s.

REMARK. As we remarked in Part I, our definition of path differs from Andrews' [And81] and Bibel's [Bib81] definition so as to demonstrate the relationship between the matrix methods and tableau/sequent methods. Each clause in the definition, when interpreted as operating on the set associated with the path, corresponds roughly to an application of an inverted sequent inference rule (see Chapter 4). This was explained in detail in Chapter 3 for the case of classical logic. (END OF REMARK.)

EXAMPLE.

$$S(\{ a_0 \}) = \{ a_0 \}$$

$$S\left(\left\{ a_3, a_9, a_{12}^1 \right\}\right) = \left\{ a_0, a_1, a_2, a_3, a_9, a_{10}^1, a_{11}^1, a_{12}^1 \right\}$$

$$S\left(\left\{ a_3, a_6^1, a_7^1, a_8^{11}, a_9, a_{12}^1 \right\}\right) = \{a_0, a_1, a_2, a_3, a_4^1, a_5^1, a_6^1, a_7^1, a_8^{11}, a_9, a_{10}^1, a_{11}^1, a_{12}^1\}$$

$$S\left(\left\{ a_3, a_6^2, a_7^2, a_8^{21}, a_6^1, a_7^1, a_8^{11}, a_9, a_{12}^1 \right\}\right) = \left\{a_0, a_1, a_2, a_3, a_4^2, a_5^2, a_6^2, a_7^2, a_8^{21}, a_4^1, a_5^1, a_6^1, a_7^1, a_8^{11}, a_9, a_{10}^1, a_{11}^1, a_{12}^1\right\}$$

(END OF EXAMPLE.)

In a similar vein we define, $D(s)$, the set of positions *dominated* by the path s. Informally, $D(s)$ represents those positions that can possibly be "reached" by further reductions. Formally, the definition of $D(s)$ is quite complex. First we define an intermediate set $I(s)$ inductively as follows: $I(s)$ is the smallest set of positions such that $s \subseteq I(s)$, and closed under the rule:

if $u \in I(s)$ and v is a child of u with $v \notin S(s)$, then $v \in I(s)$.

$D(s)$ is then defined by:

$$D(s) \stackrel{\text{df}}{=} I(s) \setminus s.$$

EXAMPLE.

$$D(\{ a_0 \}) = \text{every position of the indexed tree.}$$

$$D\left(\left\{ a_3, a_9, a_{12}^1 \right\}\right) = \left\{a_4^1, a_5^1, a_6^1, a_7^1, a_8^{11}, a_4^2, a_5^2, a_6^2, a_7^2, a_8^{21},\right.$$

$$D\left(\left\{ a_3, a_6^1, a_7^1, a_8^{11}, a_9, a_{12}^1 \right\}\right) = \left\{a_4^2, a_5^2, a_6^2, a_7^2, a_8^{21}\right\}$$

$$D\left(\left\{ a_3, a_6^2, a_7^2, a_8^{21}, a_6^1, a_7^1, a_8^{11}, a_9, a_{12}^1 \right\}\right) = \emptyset$$

Notice that the positions a_{13}^1 and a_{14}^1 are not elements of $D(s)$ after the reduction of a_{10}^1 which is of β-type. In the last path above, for example, a_9 is such that $a_9 \ll a_{13}^1$. But a_9 is of ν-type, and its child u_0 for which $u_0 \trianglelefteq a_{13}^1$ is the position a_{10}^1. But a_{10}^1 is in the set associated with the path (*i.e.*, it has been reduced already), hence a_{13}^1 is not in the set dominated by the path. (Similarly for a_{14}^1 of course.) (END OF EXAMPLE.)

A path, s, through X^μ is an *atomic path* just in case $D(s) = \emptyset$. From these definitions we conclude:

FACT 6.1 *If s is an atomic path, then for $k^\kappa \in s$ either:*

(a) k is an atomic position; or

(b) k is of γ-type, k_1 its child, and for all j, $1 \leq j \leq \mu_Q(k_1)$, $k_1^{\kappa j} \in S(s)$; or

(c) k is of ν-type, k_1 its child, and for all j, $1 \leq j \leq \mu_M(k_1)$, $k_1^{\kappa j} \in S(s)$.

EXAMPLE. Consider the previous examples of paths and their associated sets. The third path: $\{a_3, a_6^1, a_7^1, a_8^{11}, a_9, a_{12}^1\}$ consists only of atomic positions

and γ and ν-type positions. It is not atomic since not all of the children of a_3 are elements of $S(s)$. In other words, we can reduce the path further. The last path, however, is atomic, since all such reductions have been performed. (END OF EXAMPLE.)

The above definitions are somewhat complex. Luckily, the "matrix" characterisations get their name from a visual method of identifying the atomic elements of atomic paths through indexed formulae. The matrix representation of a formula described for classical logic in Chapters 1 and 3 carries over to the modal case with no change. Consider our example (signed) formula:

$$\langle \Diamond\Diamond \forall x\,(\Diamond Px \wedge \Box Qx) \Rightarrow \Diamond(\forall y Py \wedge \forall z Qz), 0\rangle$$

indexed as in Figure 6-3. If we distinguish its α-type subformulae from its β-type subformulae by placing the components of the former side-by-side and the components of the latter one above the other, we obtain a nested *matrix* thus:

$$\Diamond\Diamond^{\forall x}\left(\begin{array}{cc} \left(\Diamond\,(Px^1) \wedge \Box\,(Qx^1)\right) & \left(\Diamond\,(Px^2) \wedge \Box\,(Qx^2)\right) \end{array}\right) \Rightarrow \Diamond^{\begin{pmatrix} \forall y\,(Py^1) \\ \wedge \\ \forall z\,(Qz^1) \end{pmatrix}}$$

(For the reader's convenience we have left the individual variables in the atoms and simply indexed them x^1, x^2 to distinguish the different instances. Strictly speaking, Px^1 below should read Pa_4^1 *etc.*) Notice that the two instances of the subformula $Px \wedge Qx$ are considered to be the components of an implicit α-type formula. This follows from the γ clause (4) of the definition of paths above. If we omit the connectives, quantifiers and modal operators we are left with the skeleton matrix:

$$\begin{array}{cc} (Px^1 & Qx^1) \end{array} \quad \begin{array}{cc} (Px^2 & Qx^2) \end{array} \quad \begin{pmatrix} Py^1 \\ \\ Qz^1 \end{pmatrix}$$

which corresponds roughly to the so-called "deep formula" in the expansion tree approach of Miller [Mil84].

The atomic elements of an atomic path are simply the horizontal matrix paths through such a matrix. In this case there are two atomic paths through the formula, one with atomic elements $\{Px^1, Qx^1, Px^2, Qx^2, Py^1\}$ and one

whose atomic elements are $\{Px^1,\ Qx^1,\ Px^2,\ Qx^2,\ Qz^1\}$. More precisely, we should express these sets as positions thus:

$$\{a_6^1, a_8^{11}, a_6^2, a_8^{21}, a_{12}^1\} \quad \text{and} \quad \{a_6^1, a_8^{11}, a_6^2, a_8^{21}, a_{14}^1\}.$$

A ***connection*** in an (indexed) formula is a subpath of a path through the formula consisting of two atomic positions of different polarities, but labelled by an atomic formula with the same predicate symbol. A set of connections is said to ***span*** the formula just when every atomic path through it contains a connection from the set.

For example, the two connections $\{a_6^1, a_{12}^1\}$ and $\{a_8^{(11)}, a_{14}^1\}$ span the indexed formula displayed above. So does the connection pair $\{a_6^1, a_{12}^1\}$ and $\{a_8^{(21)}, a_{14}^1\}$.

6.2.4 Complementarity.

The generalisation of the notion of formula tree, position, path and connection to the modal language was quite straightforward. Indeed, up to this point, the modal operators have been treated simply as a new form of quantifier. As we discussed in the introduction, our aim is to formulate a (logic-dependent) notion of complementarity for connections which ensures that the existence of a spanning set of complementary connections in an (indexed) signed formula, $\langle A, 0\rangle^\mu$, entails the existence of a proof of the sequent: $\longrightarrow A$, in a sequent calculus for that logic. The correctness of the sequent system then ensures the validity of A. Conversely, since every valid formula has a sequent proof, we require that if A is valid, then there is such a spanning set of complementary connections in $\langle A, 0\rangle^\mu$ for some multiplicity μ; *i.e.*, the matrix characterisations are complete. The formulation of complementarity should be structural; *i.e.*, it should be in terms of syntactic properties of the formula A.

In this section we present appropriate definitions of complementarity for connections for the modal logics under consideration. The propositional fragments of the logics are treated first, the first-order systems second. In §6.3 and §6.4

we prove the correctness and completeness of the resulting characterisations of validity.

6.2.4.1 Propositional modal logics.

For a formula A of pure propositional logic, connections are complementary *by definition.* Since there is no need for multiplicities (no modal operators or quantifiers) we obtain the simple characterisation of validity (re)derived in Part I.

THEOREM 6.2 (ANDREWS [AND81], BIBEL [BIB81]) *A propositional formula A is valid iff there exists a set of connections that spans $\langle A, 0\rangle$.*

As discussed in Part I complementarity in propositional logic is this simple because there is no order dependence between the sequent inference rules. In the last chapter we showed how the modal inference rules induced such an order dependence. Thus, in the presence of modal operators we must be more careful. The propositional complementarity of two atoms is not enough to guarantee that a derivation of the endsequent exists with that connection at one of its leaves. We saw that the number and nature of the modal operators that quantify atoms are crucial factors in determining whether the appropriate sequent derivation exists. We capture this notion of "modal context" in terms of the formula tree as shown below.

The following definitions are introduced for a given (indexed) formula tree for a given (indexed) formula X^{μ}. Let $T_M(\mu)$ denote the union of $\mathcal{V}_0(\mu)$ and $\Pi_0(\mu)$. We associate with each position u of the formula tree a sequence of positions, $\mathrm{pre}(u)$, called a *prefix*, as follows: if $u_1 \ll u_2 \ll \cdots \ll u_n \trianglelefteq u$, $1 \leq n$, are those elements of $T_M(\mu)$ that dominate u in the formula tree, then

$$\mathrm{pre}(u) = \begin{cases} u_1 u_2 \cdots u_n, & \text{K, K4, D, D4, T, S4;} \\ u_n, & \text{S5.} \end{cases}$$

The prefix of a position encodes the modal context of the position within the formula tree. For example, the (non-S5) prefix of a_6^1 in the indexed formula of Figure 6–3 is the sequence $a_0 a_2 a_3 a_6^1$, while the (non-S5) prefix of a_{12}^1 is $a_0 a_{10}^1$. The S5-prefix of the former position is a_6^1, and of the latter a_{10}^1.

We shall use p and q as metavariables for prefixes, and $P(\mu)$ to denote the complete set of prefixes of the positions of X^{μ}. We shall use $p \prec q$ to denote that p is a proper initial sequence of q, and $u \in p$ to denote that u is a unit element (*i.e.*, $u \in T_M(\mu)$) of the sequence p.

The definition of prefixes gives us the following fact:

FACT 6.3 *For all positions u and v, $v \in pre(u)$ implies $v \not\preceq u$.*

Let T^+ denote the set of sequences (words) generated from some set T of elements, and $T^* = T^+ \cup \{\emptyset\}$. We define $\mathcal{L}$-accessibility relations on $T^* \times T^*$ by the conditions shown in Table 6–1. For example, the S4-accessibility relation on T^* is *defined* as follows: for $p, q \in T^*$, $p\ R_0\ q$ if and only if either:

(a) (general) $q = pu$, where $u \in T$; or

(b) (reflexive) $q = p$; or

(c) (transitive) $p \prec q$.

An equivalent definition is: for all $p, q \in T^*$,

$$p\ R_0\ q \quad \text{iff} \quad p \preceq q.$$

We prefer the first since we can generate the definitions for all of the logics (except S5) by means of the general, reflexive and transitive conditions. Notice that the transitive condition subsumes the general condition. The definitions for the logics under consideration are given in Table 6–2. With these definitions we immediately have the following fact:

FACT 6.4 *Let $\mathcal{L}$ be one of the logics: K, K4, D, D4, T, S4. Let R_0 be the $\mathcal{L}$-accessibility relation on T^*. Then, for all $p, q \in T^*$,*

$$p\ R_0\ q \quad \textit{implies} \quad p \preceq q.$$

Property	Condition
general	$p\ R_0\ pu, \quad p \in T^*, u \in T$
reflexive	$p\ R_0\ p, \quad p \in T^*$
transitive	$p\ R_0\ pq, \quad p \in T^*, q \in T^+$

Table 6–1: Prefix conditions.

$\mathcal{L}$	Properties of R_0
K, D	general
T	general, reflexive
K4, D4	general, transitive
S4	general, reflexive, transitive
S5	$u\ R_0\ v, \quad u, v \in T$

Table 6–2: Accessibility relations on prefixes.

REMARK. These definitions are adapted from the so-called *prefixed* tableau systems of Fitting [Fit72,Fit83]. We discuss the relationship between these systems and the matrix systems in detail in Chapter 8. (END OF REMARK.)

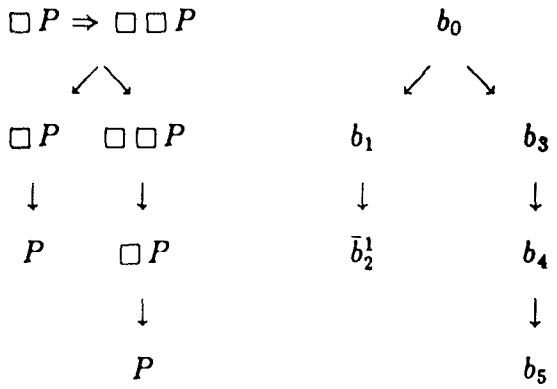
Of course, for our purposes, the sequences are generated by the set $T_M(\mu)$. We can think of prefixes either semantically or proof-theoretically. Semantically, each prefix is the name of a point in a model. Accessibility relations on prefixes are used to represent properties of the accessibility relation of the model. Proof-theoretically, the prefix of a position represents the sequence of (inverted) modal rule applications that are necessary to produce an occurrence of the subformula rooted at that position as an S-formula in a derivation. Recall that our aim is to guarantee that, given a connection, there exist a sequence of (inverted) rule applications that produce images of both the atomic formulae that label the components of the connection in the same sequent. But the (inverted) π rules rules cause formulae to be "deleted" from the sequent. On the other hand, in certain logics, ν-type formulae may be preserved through such a deletion operation. We must ensure that the ancestor formulae of the pair of atoms that form the connection are not "deleted."

Let us return to the example given in the introduction, namely the S4-validity of $\Box P \Rightarrow \Box\Box P$. An indexed formula tree for the signed formula: $\langle \Box P \Rightarrow \Box\Box P, 0\rangle$, with a constant multiplicity of 1, is shown in Figure 6–7. Notice that we have distinguished the ν_0-type element, b_2, with an overbar: $\bar{b}_2$.

Recall the S4-proof of this formula:

$$
\begin{array}{lll}
w_2 & P \longrightarrow P & \\
w_2 & \Box P \longrightarrow P & \Box \longrightarrow \quad (\nu) \\
w_1 & \Box P \longrightarrow \Box P & \longrightarrow \Box \quad (\pi) \\
w_0 & \Box P \longrightarrow \Box\Box P & \longrightarrow \Box \quad (\pi) \\
w_0 & \longrightarrow \Box P \Rightarrow \Box\Box P & \longrightarrow\Rightarrow \quad (\alpha)
\end{array}
$$

where we have included the (notional) point about which each sequent makes an assertion. We have that $w_0\ R\ w_1$ and $w_1\ R\ w_2$. The next step is to include the



u	pol(u)	lab(u)	pre(u)	pre$_{S5}$(u)	Ptype(u)	Stype(u)
b_0	0	$\Box P \Rightarrow \Box\Box P$	b_0	b_0	α	π_0
b_1	1	$\Box P$	b_0	b_0	ν	α_1
$\bar{b}_2^1$	1	P	$b_0\bar{b}_2^1$	$\bar{b}_2^1$	–	ν_0
b_3	0	$\Box\Box P$	b_0	b_0	π	α_2
b_4	0	$\Box P$	b_0b_4	b_4	π	π_0
b_5	0	P	$b_0b_4b_5$	b_5	–	π_0

Figure 6–7: Indexed formula tree for: $\langle \Box P \Rightarrow \Box\Box P, 0\rangle$.

prefixes of the S-formulae in the derivation thus:

$$
\begin{array}{lll}
w_2 & \dfrac{P_{b_0\bar{b}_2^1} \longrightarrow P_{b_0b_4b_5}}{(\Box P)_{b_0} \longrightarrow P_{b_0b_4b_5}} & \Box \longrightarrow \ (\nu) \\
w_2 & \dfrac{(\Box P)_{b_0} \longrightarrow P_{b_0b_4b_5}}{(\Box P)_{b_0} \longrightarrow (\Box P)_{b_0b_4}} & \longrightarrow \Box \ (\pi) \\
w_1 & \dfrac{(\Box P)_{b_0} \longrightarrow (\Box P)_{b_0b_4}}{(\Box P)_{b_0} \longrightarrow (\Box\Box P)_{b_0}} & \longrightarrow \Box \ (\pi) \\
w_0 & \dfrac{(\Box P)_{b_0} \longrightarrow (\Box\Box P)_{b_0}}{\longrightarrow (\Box P \Rightarrow \Box\Box P)_{b_0}} & \longrightarrow\Rightarrow \ (\alpha)
\end{array}
$$

This is an S4-proof because we can preserve the antecedent image $(\Box P)_{:b_0}$ through two applications of the S4 (inverted) π rule. This in turn is permitted because the accessibility relation for S4 is transitive, *i.e.*, $w_0 \; R \; w_2$.

We come now to the crucial issue. Rather than dealing in terms of notional points of a model, we deal instead with prefixes. We use the prefixes of subformulae to denote the notional points. The properties of the accessibility relation on points is represented by the accessibility relation on prefixes given for S4 in Tables 6–1 and 6–2.

With this as the aim, we are forced to consider the following correspondence between prefixes and points:

$$
\begin{aligned}
b_0 &\longleftrightarrow w_0 \\
b_0b_4 &\longleftrightarrow w_1 \\
b_0b_4b_5 &\longleftrightarrow w_2
\end{aligned}
$$

Now consider the prefix $b_0\bar{b}_2^1$ of the atomic antecedent image of P in the above derivation. To what notional point does it correspond? We have a choice. Due to the transitivity of the accessibility relation for S4, $b_0\bar{b}_2^1$ could denote *any* point accessible from b_0 (*i.e.*, w_0). By the entries in Tables 6–1 and 6–2 for S4 this means any prefix of which b_0 is an intial sequence. Consequently, we can choose to let $b_0\bar{b}_2^1$ denote the point w_2 by the prefix mapping:

$$\bar{b}_2^1 \; \mapsto \; b_4b_5$$

Turning the argument about, roughly speaking, provided we can find a mapping of ν_0 positions, to $T_M(\mu)^*$ such that the prefixes of the positions of the connection (in this case $b_0\bar{b}_2^1$ and $b_0b_4b_5$) are identical, the connection is S4-complementary.

Looking at it another way, the prefix mapping encodes the range over which the (inverted) ν rule must not be applied to the S-formula $(\Box P)\cdot_{b_0}$ in order to preserve this ancestor of the atom $P_{b_0\bar{b}_2^1}$.

The discussion above motivates the following definitions. Let σ_M be a mapping from $\mathcal{V}_0$ to $T_M(\mu)^*$. Recall that $\mathcal{P}(\mu)$ is the set of prefixes of the indexed formula X^μ. Define $\mathcal{P}_{\sigma_M}(\mu)$ to be the image of $\mathcal{P}(\mu)$ under σ_M in the following sense:

$$\mathcal{P}_{\sigma_M}(\mu) \stackrel{\text{df}}{=} \left\{ p \mid p \preceq \sigma_M^{\#}(q)\,,\, q \in \mathcal{P}(\mu) \right\},$$

where $\sigma_M^{\#}\colon T_M(\mu)^* \to T_M(\mu)^*$ is the homomorphic extension of σ_M to $T_M(\mu)^*$.

Such a mapping σ_M is a *modal substitution* just in case:

A. $u \,\varepsilon\, \sigma_M(v)$ implies $\sigma_M(u) = u$.

B. $pu \in \mathcal{P}_{\sigma_M}(\mu)$ implies $pu = \sigma_M^{\#}(\mathrm{pre}(u))$.

The first condition is for technical convenience only, and restricts our attention to substitutions that are "minimal" in an obvious sense. The second condition ensures the coherence of the substitution and reflects the fact that we shall compute modal substitutions by unifying the prefixes of positions labelled with atomic formulae. We shall point out where these conditions are utilised in the correctness proofs.

A modal substitution σ_M induces an equivalence relation $\sim_M$ and a relation $\sqsubset_M$ on $T_M(\mu) \times T_M(\mu)$ as follows:

1. If $\sigma_M(u) = v$ and $v \in \mathcal{V}_0(\mu)$, then $u \sim_M v$.

2. If $\sigma_M(u) = p$ and $p \notin \mathcal{V}_0(\mu)$, then for all $v \preceq p$, $v \sqsubset_M u$.

3. If $v \sqsubset_M u$ and $u \sim_M u'$, then $v \sqsubset_M u'$.

This definition should be compared with the similar one given for classical quantifiers in Chapter 3.

A modal substitution σ_M is *$\mathcal{L}$-admissible* provided

1. σ_M respects the $\mathcal{L}$-accessibility relation R_0 on $T_M(\mu)^*$; *i.e.*, for all $p, q \in T_M(\mu)^*$,

$$p\, R_0\, q \quad \text{implies} \quad \sigma_M^{\#}(p)\, R_0\, \sigma_M^{\#}(q)\,.$$

2. (K-logics only) $u \sim_M u'$ implies $v \sqsubset_M u$ (and hence $v \sqsubset_M u'$) for some α-related position v.

3. $\lhd = (\ll \cup \sqsubset_M)^+$ is irreflexive.

Once again the reader should compare this definition with the definition of admissibility for a classical mapping. Except for the condition pertaining to the K-logics the only addition is the first condition concerning the stability of $\mathcal{L}$-accessibility relations on prefixes.

Two positions are α-related just in case there is some path s for which both positions are elements of $S(s)$. Alternatively, a position u is α-related to a position u' just in case the $\ll$-greatest common ancestor of u and u' is not a β-position. By "$\ll$-greatest common ancestor of u and u'" we mean a position v such that $v \ll u$, $v \ll u'$, and there is no position v', distinct from v, with $v \ll v'$, such that $v' \ll u$ and $v' \ll u'$ both hold. In terms of matrices, positions that are not α-related appear in the same column vertically separated.

The appropriate notion of complementarity for the propositional modal logics under consideration is as follows: if σ_M is an $\mathcal{L}$-admissible modal substitution for X^μ, a connection $\{ u, v \}$ in X^μ is said to be *σ_M-complementary* iff

$$\sigma_M^{\#}(\mathrm{pre}(u)) = \sigma_M^{\#}(\mathrm{pre}(v))\,.$$

REMARKS. The relation $v \lhd u$ should be interpreted as a prescription that "position v should be introduced into a path (by the reduction of its parent) before position u." Accordingly we call the relation the *reduction ordering*. Its

irreflexivity ensures that we could construct a sequent proof with $\longrightarrow A$ as root using the sequent systems of Chapter 4 (assuming that the signed formula X is $\langle A, 0 \rangle$). This last remark will be made clear in the correctness proofs of §6.3.

Suitable substitutions can be computed using variants on a string-unification algorithm. For all of the logics under consideration the set of most general unifiers is finite, but not necessarily a singleton. We discuss this further in the next chapter. For S5 the standard unification algorithm suffices. The admissibility check is an check for acyclicity if $\lhd$ is interpreted as a directed graph.

The extra condition for the K-logics reflects the fact that there is no rule which, when inverted, reduces ν-type formulae only. Reductions of ν-type S-formulae in a sequent derivation only occur during the reduction of π-type formulae. We must therefore ensure that every ν_0 position (formula) is associated, via the modal substitution, to a π_0 position (formula) whose reduction causes the introduction of the ν_0 position (formula). The condition stated above ensures that this is indeed the case. We shall indicate in the correctness proofs where this condition plays its rôle. (END OF REMARKS.)

We have defined what it means for a connection to be complementary in a modal logic. The final step is to characterise the validity of a modal formula in terms of such complementary connections. This is achieved by extending the complementarity of a connection to the paths that contain it. Informally, this is the final link in the correspondence between the modal sequent calculi of Chapter 4 and the matrix methods: a path made complementary itself by containing a complementary connection as a subpath corresponds to an instance of the basic sequent.

Let σ_M be an $\mathcal{L}$-admissible modal substitution for X^μ. A path, s, through X^μ is said to be σ_M-complementary just in case:

1. It contains a σ_M-complementary connection $\{u, v\}$ (as a subpath).

2. (K-logics only.) For all $u' \varepsilon \sigma(\mathrm{pre}(u))$, $u' \in S(s)$.

A set of σ_M-complementary connections *spans* X^μ just in case the set ensures that every atomic path through X^μ is σ-complementary.

REMARK. For the idealisable modal logics: D, D4, T, S4, S5, this mirrors the definition for classical logic given in Part I. For the non-idealisable K-logics: K and K4, an extra condition is imposed. This condition can be motivated by reference to the sequent calculi of Chapter 4. Informally, since there is no ν rule in the sequent calculi for the K-logics, every reduction of a ν-type formula (position) must be "driven" by the reduction of a π-type formula (position). This constraint is met in the matrix methods by the extra condition (2) on K and K4-admissible substitutions, which force such "linking" between ν_0 and π_0 positions. But such connections only correspond to basic sequents that can be "constructed," so to speak, by (inverted) applications of the π rule. The definition of path was purely structural, and hence the same for each logic. This is desirable so that path checking algorithms developed for classical logic (*eg.*, [Bib82a,HB82]) are immediately applicable to the modal logics. For the K-logics however, although a path may *structurally* contain a complementary connection (as a subpath), that connection may not in fact be "constructable" without an independent ν rule. The extra condition is necessary to ensure that a path is only classified as being complementary if the complementary connection it contains is constructable in this sense. We shall remark on this condition in the correctness proofs below where these notions are made more precise. (END OF REMARK.)

In §6.3 and §6.4 we prove both halves of the following theorem:

THEOREM 6.5 *A propositional modal formula A is $\mathcal{L}$-valid iff there is a modal multiplicity μ_M, an $\mathcal{L}$-admissible modal substitution σ_M and a set of σ_M-complementary connections that spans the indexed formula $\langle A, 0\rangle^{\mu_M}$.*

EXAMPLE. Consider our example formula

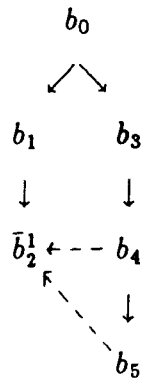
$$\langle \Box P \Rightarrow \Box\Box P, 0\rangle$$

and its formula tree in Figure 6–7. The connection $\left\{ \bar{b}_2^1, b_5 \right\}$ spans the formula. The prefixes of these atomic positions are: $\mathrm{pre}\left(\bar{b}_2^1\right) = b_0\bar{b}_2^1$ and $\mathrm{pre}(b_5) = b_0b_4b_5$

(where we distinguish ν_0-type positions in prefixes with an overbar for the reader's convenience). Consider the mapping σ_M defined as follows:

$$\sigma_M(u) = \begin{cases} b_4 b_5, & u = \bar{b}_2^1 \\ u, & \text{otherwise.} \end{cases}$$

Under σ_M the two prefixes are identical. Firstly, σ_M is a modal substitution since Conditions A and B hold. (Condition B is easy to check: simply notice that the only element of $P(\mu)$ affected by the substitution is $b_0\bar{b}_2^1$, and that its image under σ_M is $b_0 b_4 b_5$. Furthermore, $\sigma_M^{\#}(\text{pre}(b_5)) = b_0 b_4 b_5$ and $\sigma_M^{\#}(\text{pre}(b_4)) = b_0 b_4$.) Secondly, σ_M respects $\mathcal{L}$-accessibility relations on prefixes that satisfy the transitive condition. Thirdly, the K-condition is satisfied since b_4 and b_5 are α-related to $\bar{b}_2^1$. (Indeed there are no β-type positions in the formula tree at all.) Finally, the reduction ordering $\lhd$ induced by the mapping can be pictured as a directed graph as follows:



where we have used dotted arrows to represent the $\sqsubset_M$ relation. Since the directed graph is acyclic we conclude that the mapping is K4, D4 and S4-admissible. This suffices to show that the path is σ_M-complementary for D4 and S4, and therefore that the formula is valid in these logics. For K4 we have both b_4 and b_5 are elements of the set associated with the single atomic path through the formula. Consequently, this path is σ-complementary and the formula is also valid in K4.

For S5 the prefixes are $\text{pre}(b_2) = \bar{b}_2^1$ and $\text{pre}(b_5) = b_5$. The mapping that takes $\bar{b}_2^1$ to b_5 is clearly S5-admissible and its reduction ordering is a subgraph of that shown above (and hence acyclic also). The formula is therefore S5-valid also. (END OF EXAMPLE.)

6.2.4.2 First-order modal logics.

Extending the matrix characterisations of validity in the propositional fragments to the quantified logics is straightforward. We consider constant, varying and cumulative domain variants of the quantified logics. The notion of a *first-order* substitution below is identical to that introduced in Part I for classical logic.

Let $T_Q(\mu)$ denote the set $\Gamma_0(\mu) \cup \Delta_0(\mu)$ (cf. $T_M(\mu)$). A first-order substitution is a mapping $\sigma_Q: \Gamma_0(\mu) \mapsto T_Q(\mu) \cup C$, where C is the set of constants in the formula being tested for validity.

For constant domains, a connection is interpreted as complementary if we can find a first-order substitution σ_Q that render the (atomic) labels of the two atomic positions identical.

For varying and cumulative domains, the modalities and quantifiers interact. Universally quantified variables only range over those individuals that "exist" at the point denoted by the prefix of their quantifiers. Existential quantifiers express the existence of individuals only in the point denoted by their prefixes. Consequently, our first-order substitution σ_Q must respect the modal substitution σ_M.

For soundness, we must place restrictions on first-order substitutions to ensure that the positions representing parameters introduced for existentially bound variables (replaced in the labels of positions by elements of $\Delta_0(\mu)$) are indeed arbitrary. In terms of sequent systems, we must ensure that such positions are introduced (by the reduction of their parent) before the introduction of any position representing a universally bound variable (*i.e.*, element of $\Gamma_0(\mu)$) which is to receive the same parameter. These notions should be familiar from Part I. The similarity between the restrictions on quantifier reductions and the restrictions on modal operator reductions is not accidental [Smu70].

A first-order substitution $\sigma_Q: \Gamma_0(\mu) \rightarrow T_Q(\mu) \cup C$ induces an equivalence relation $\sim_Q$ and a relation $\sqsubset_Q$ on $T_Q \times T_Q$ as follows:

1. If $\sigma_Q(u) = v$ and $v \in \Gamma_0$, then $u \sim_Q v$.

2. If $\sigma_Q(u) = v$ and $v \notin \Gamma_0$, then $v \sqsubset_Q u$.

3. If $v \sqsubset_Q u$ and $u \sim_Q u'$, then $v \sqsubset_Q u'$.

A *combined* substitution is a pair consisting of a modal substitution and a first-order substitution. A combined substitution $\langle \sigma_M, \sigma_Q \rangle$ is $\mathcal{L}$-admissible provided

1. σ_M respects $\mathcal{L}$-accessibility relations, as before.

2. (K-logics only) $u \sim_M u'$ implies $v \sqsubset_M u$ (and hence $v \sqsubset_M u'$) for some α-related position v.

3. $\lhd = (\ll \cup \sqsubset_M \cup \sqsubset_Q)^+$ is irreflexive.

4. The following condition holds for σ_M and σ_Q depending on the domain condition for the logic $\mathcal{L}$:

   Constant domains: No condition.

   Varying domains: If $\sigma_Q(u') = v'$, then $\sigma_M^{\#}(\mathrm{pre}(v')) = \sigma_M^{\#}(\mathrm{pre}(u'))$.

   Cumulative domains: If $\sigma_Q(u') = v'$, then either:

   (a) $\sigma_M^{\#}(\mathrm{pre}(v')) = \sigma_M^{\#}(\mathrm{pre}(u'))$; or

   (b) $\sigma_M^{\#}(\mathrm{pre}(v'))\ R_0\ \sigma_M^{\#}(\mathrm{pre}(u'))$.

REMARKS. There is no coupling of the modal and first-order substitutions for constant domains. For varying domains however, there is such a coupling, namely: "parameters" (elements of $T_Q(\mu)$) associated by the first-order substitution must have identical prefixes under the modal substitution. (Intuitively, the prefix of such an element is the point at which it exists.) The corresponding coupling for cumulative domains is weaker: the prefix of the "variable parameter" (element of Γ_0) need only be $\mathcal{L}$-accessible from the prefix of its image. (END OF REMARKS.)

The appropriate notion of complementarity is then as follows: if σ is an $\mathcal{L}$-admissible combined substitution for X^μ, a connection $\{ u, v \}$ in X^μ is σ-complementary iff

1. $\sigma_M^{\#}(\mathrm{pre}(u)) = \sigma_M^{\#}(\mathrm{pre}(v))$.

2. $\sigma_Q(\mathrm{lab}(u)) = \sigma_Q(\mathrm{lab}(v))$.

REMARK. Recall that the label of an atomic position u is an atomic formula with elements of $T_Q(\mu)$ in place of what otherwise would be free individual variables. This is a consequence of the definition of the label of an indexed position given in §6.2.2. Consequently substitutions can be extended to modal formulae in the obvious way. This makes sense of the second condition above. (END OF REMARK.)

A path is σ-complementary under the same conditions as were discussed in the previous section. The difference between the first-order and propositional characterisations of validity is captured solely in the notion of complementarity for connections.

In §6.3 and §6.4, we prove both halves of the theorem:

THEOREM 6.6 *A (first-order) modal formula A is $\mathcal{L}$-valid iff there is a multiplicity μ, an $\mathcal{L}$-admissible combined substitution σ and a set of σ-complementary connections that spans the indexed formula $\langle A, 0\rangle^{\mu}$.*

EXAMPLE. Consider the example formula

$$\langle \Diamond\Diamond \forall x\, (\Diamond Px \wedge \Box Qx) \Rightarrow \Diamond(\forall y Py \wedge \forall z Qz), 0\rangle$$

indexed as in Figure 6–3. We noted before that the connections $\{a_6^1, a_{12}^1\}$ and $\{a_8^{21}, a_{14}^1\}$ span the formula. The prefixes and labels of positions are also show in the figure. The mapping σ_M that takes $\bar{a}_{10}^1$ to $a_2 a_3 a_6^1$, $\bar{a}_8^{21}$ to a_6^1, and is the identity everywhere else, unifies the respective pairs of prefixes. It is easy to check that σ_M is *propositionally* admissible for the transitive logics K4, D4 and S4.

The mapping σ_Q that takes $\bar{a}_4^1$ to a_{12}^1 and $\bar{a}_4^2$ to a_{14}^1, and is the identity everywhere else, unifies the labels of the atomic positions comprising the connections.

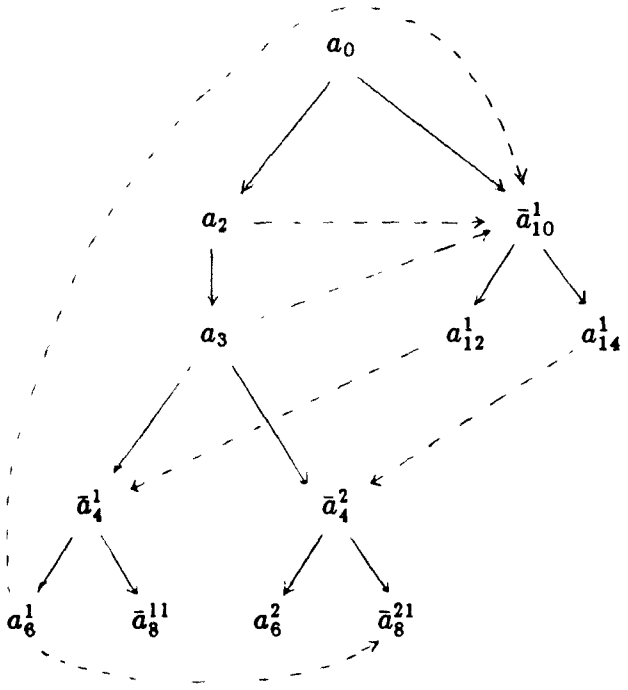


Figure 6–8: Reduction order for connections.

The reduction ordering induced by the combined substitution $\langle \sigma_M, \sigma_Q \rangle$ is shown in Figure 6–8. Notice that the graph is cyclic. The two connections cannot be simultaneously complementary in any of the first-order logics. Recall that in Chapter 5 we presented this formula for which we could not find a proof. We ascribed the problem to an interaction between the reduction order for the modal operators with the reduction order for quantifiers. Here we see how the cyclicity of the reduction ordering $\lhd$, induced by the combined substitution, captures such constraints. More examples can be found in the next chapter. (END OF EXAMPLE.)

6.3 Correctness.

In this section we prove that the matrix characterisations of validity presented above are correct for the modal logics under consideration. The method used is based on standard techniques for systematic correctness proofs of analytic tableau (*eg.*, [Smu68,Fit83]) and cut-free sequent calculi (*eg.*, [Kle68]), together with the techniques developed in Part I for classical logic.

For the rest of this section we assume the following: let A be a modal sentence whose validity we are interested in, X the signed modal formula $\langle A, 0\rangle$, μ a multiplicity for X, and σ a combined $\mathcal{L}$-admissible mapping for X^μ. Under these assumptions, the statement of correctness for the matrix characterisations is:

> if there is a set of σ-complementary connections that span $\langle A, 0\rangle^\mu$, A is $\mathcal{L}$-valid.

By "position" we mean "position of X^μ," by "path" we mean "path through X^μ," *etc.* In addition we shall not refer explicitly to the components, σ_M and σ_Q, of the combined substitution, but use σ to denote both, leaving it to the context to determine which we mean. Since the domains and codomains of both components are disjoint, no confusion can arise. Furthermore we shall not distinguish between σ and its homomorphic extension over prefixes $\sigma^\#$, formulae and signed formulae, again leaving it up to the context to determine which we mean. First we provide an overview of the ensuing proofs.

6.3.1 Overview.

Recall that a path s determines an associated set of positions $\mathcal{S}(s)$. $\mathcal{S}(s)$ represents the positions that dominate the elements of s in the formula ordering $\preceq$. Recall also the dual notion $\mathcal{D}(s)$, that represents the set of distinct positions

that can potentially be "reached" by further path reductions. In particular, recall that $S(s) \cap D(s) = \emptyset$, by definition.

$S(s)$ represents a sequent in a certain sense. We called the relation $\lhd$ the "reduction ordering," when it was introduced above. We shall show (§6.3.2, Proposition 6.14) that, starting from the singleton set of paths containing only the root path, a sequence of reduction operations can be performed on this set of sets — replacing a path by the paths obtained from its reduction — until we are left only with the irreducible set of atomic paths through X^u. Moreover, this can be done in such a way that at none of these reductions is a position introduced into a path whilst a $\lhd$-lesser position still remains reachable by further reductions. This defines informally what we mean by a *proper* reduction. The paths are said to be $\lhd$-*compatible*. In other words, the order in which positions are introduced by reduction respects the reduction ordering $\lhd$. The irreflexivity (acyclicity) of $\lhd$ is the crucial condition that guarantees the existence of an appropriate sequence of proper reductions.

Intuitively, for a subclass of the logics, proper reductions correspond to applications of the sequent inference rules on the sequent represented by the path. We show that each proper reduction is correct in a sense reminiscent of the correctness of the individual sequent rules of Chapter 4. Consequently, Proposition 6.14 ensures the existence of a sequent derivation of $\longrightarrow A$ with the atomic paths corresponding to its leaves. The existence of a spanning set of complementary connections entails that all the leaves of this derivation are instances of the basic sequent, *i.e.*, the derivation is a proof of $\longrightarrow A$, and consequently A is valid.

Our proofs of the correctness of proper path reductions are semantic, rather than proof-theoretic, in part because we have not developed suitable sequent calculi for S5 and the varying and constant domain versions of the logics under consideration. More centrally, we believe that this approach will support the application of the ideas of this chapter to other (more complicated) logics for which, like S5 and the constant domain modal logics, standard, cut-free sequent systems are not available.

The formal ideas are based on Fitting's justification of his prefixed tableau systems [Fit72,Fit83]. We decompose the proofs into four sections for the reader's convenience, and to suggest the relationship with the correctness proofs for the sequent rules presented in Chapter 4. The reader is invited to review those proofs as an aid. Roughly speaking, we consider a path to be *$\mathcal{L}$-satisfiable* if the set of positions associated with it can be interpreted in an $\mathcal{L}$-model in a natural way. We show correctness by demonstrating that:

1. A path that contains a complementary connection is not $\mathcal{L}$-satisfiable (§6.3.3.1).

2. If a $\lhd$-compatible path through X^{μ} is $\mathcal{L}$-satisfiable, a proper reduction ensures that at least one of the resulting paths is also $\mathcal{L}$-satisfiable (§§6.3.3.2–6.3.3.4).

3. The non-validity of A means the root path is $\mathcal{L}$-satisfiable (§6.3.4).

Consequently, the fact that we can reduce the root path to the set of atomic paths ensures that at least one of these paths is $\mathcal{L}$-satisfiable. The existence of a spanning set of complementary connections for X^{μ} ensures that all atomic paths contain complementary connections, and hence are not $\mathcal{L}$-satisfiable, a contradiction. A must therefore be valid, and the characterisations correct.

6.3.2 Proper reductions.

For the most part, the notion of a proper reduction can be presented (but not justified) independently of a particular logic. There is an important exception, namely, proper ν-reductions.

The logics partition into two classes: the idealisable logics D, D4, T, S4, S5, and the non-idealisable logics K and K4. The condition of idealisation ensures that there is a point accessible from any given point of a frame. This structure sanctions the inclusion of a ν rule in the sequent calculi for the idealisable logics presented in Chapter 4. The form of this rule is logic dependent; a dependence

which will be reflected in the different cases considered in the justification of proper ν-reductions in the sequel. For the non-idealisable K-logics however, there is no ν rule. A ν-reduction of a path s, to obtain s', introduces a new position into $S(s')$. The absence of a ν sequent rule for the K-logics might suggest that no form of ν-reduction is justifiable. In fact, we are able to justify ν-reductions for these logics in which the prefix of the new position is already a prefix of an element of $S(s)$.

Below, we define the notion of a $\lhd$-compatible path. Roughly speaking, a proper reduction is one that preserves the $\lhd$-compatibility of paths. We capture the difference between the idealisable and non-idealisable logics via alternative definitions of $\lhd$-compatibility, and hence of proper ν-reductions.

6.3.2.1 Idealisable logics.

A position $u \in D(s)$ is said to be *unrestricted for* a path s just in case there is no $\lhd$-greater element in $S(s)$; *i.e.*, no $v \in S(s)$ such that $u \lhd v$. A path s is said to be $\lhd$-*compatible* just in case every position of $D(s)$ is unrestricted for s. An element of $D(s)$ that *is* restricted (*i.e.*, not unrestricted) in some sense "cannot be reached" by proper reductions. Some reduction has been performed "out of order" as defined by the reduction ordering $\lhd$.

We say that a non-atomic, $\lhd$-compatible path, s, is *properly reducible* on a position $u \in s$ just when the children of u introduced by the reduction are $\lhd$-least elements of $D(s)$. Notice that a proper reduction of s transfers positions from $D(s)$ to $S(s)$.

LEMMA 6.7 *The root path* $\{k_0\}$ *is* $\lhd$*-compatible.*

PROOF. Notice that $S(\{k_0\}) = \{k_0\}$, and that k_0 is the $\lhd$-least position. ■

We now come to the main lemma for the idealisable logics.

LEMMA 6.8 *If s is a non-atomic $\lhd$-compatible path, there is some element $u \in s$ on which s is properly reducible. Moreover, the paths that result from such a proper reduction are themselves $\lhd$-compatible.*

PROOF. Since s is non-atomic, $\mathcal{D}(s)$ is non-empty. The set of $\lhd$-least elements of $\mathcal{D}(s)$ is also non-empty since $\lhd$ is irreflexive (*i.e.*, acyclic when viewed as a directed graph). Consider the parents (elements of s) of these $\lhd$-least elements. We claim that all of the positions introduced by reduction on any of these parents are $\lhd$-least elements of $\mathcal{D}(s)$.

Observe that this holds trivially if one of the parents has only one child (*i.e.*, is of π or δ type), since its child is $\lhd$-least by assumption. Also, if one of the parents is generative (*i.e.*, is of ν or γ type) we may choose to introduce any of its children that are $\lhd$-least in $\mathcal{D}(s)$ (at least one such child must exist by assumption).

Suppose such a parent u is of α or β type. We claim that both the children u_1 and u_2 of u are $\lhd$-least elements of $\mathcal{D}(s)$. The secondary types of these children are one of: α_1, α_2, β_1 or β_2. Notice that there is no $v \in \mathcal{D}(s)$ with $v \ll u_i$, $i = 1, 2$, since such a v would have $v \trianglelefteq u$ (the parent of the u_i) and thus be an element of $\mathcal{S}(s)$. Also, since $u_i \notin T_M(\mu)$ and $u_i \notin T_Q(\mu)$, $i = 1, 2$, any $v \in \mathcal{D}(s)$ with $v \lhd u_i$ must have $v \lhd u$. (Another way of putting this is that the u_i do not participate in any $\sqsubset_M$ or $\sqsubset_Q$ relationships.) But this contradicts v being unrestricted and hence the $\lhd$-compatibility of s. Both children must therefore be $\lhd$-least elements of $\mathcal{D}(s)$.

Note how the condition of irreflexivity required of the reduction ordering, $\lhd$, induced by an admissible substitution, ensures that the set of $\lhd$-least elements of $\mathcal{D}(s)$ for a non-atomic path is non-empty.

We have shown the existence of an element of s, say u, all of whose children are $\lhd$-least elements of $\mathcal{D}(s)$. s is therefore properly reducible on u. Consider a path s' obtained from the proper reduction of s on u. Clearly, $\mathcal{D}(s')$ is subset of $\mathcal{D}(s)$ and therefore every element of the former is unrestricted in s by hypothesis (s is $\lhd$-compatible). Also $\mathcal{S}(s')$ is a superset of $\mathcal{S}(s)$,

containing as extra elements some or all of the children of u, depending on the type of u. Hence, a position $v \in D(s')$ could only fail to be unrestricted in s' if one of the children of u, say u_i, was such that $v \lhd u_i$. But u_i, $i = 1,2$, is a $\lhd$-least element of $D(s)$ by construction, so this cannot be. Hence s' is $\lhd$-compatible. ■

6.3.2.2 Non-idealisable logics.

For the K-logics we need variants of the notions introduced above. Define the intermediate set $I_K(s)$ inductively as follows (cf.$I(s)$): $I_K(s)$ is the smallest set of positions such that $s \subseteq I_K(s)$, and closed under the rule:

if $u \in I_K(s)$ and v is a child of u with $v \notin S(s)$, then:

(a) if v is not of ν_0-type, $v \in I_K(s)$;

(b) if v is of ν_0-type, $v \in I_K(s)$ provided: for every $v' \in \sigma(v)$, $v' \in S(s) \cup I_K(s)$.

We then define $D_K(s)$ in terms of $I_K(s)$ (cf.the definition of $D(s)$ in terms of $I(s)$):

$$D_K(s) \stackrel{\text{df}}{=} I_K(s) \setminus s.$$

As an immediate consequence of this definition we have:

FACT 6.9 *Let s be a path and $u_0 \in D_K(s)$ a position of ν_0-type. For every $v \in \sigma(u_0)$, $v \in S(s) \cup D_K(s)$.*

Clearly, $D_K(s) \subseteq D(s)$. The positions omitted from $D_K(s)$ but contained in $D(s)$ are those ν_0 type positions (and their descendants) whose prefix contains elements neither in the set associated with the path, nor "reachable" by further path reductions of the kind outlined below. We define K-atomic paths to be those paths for which $D_K(s) = \emptyset$ (cf. atomic paths where $D(s) = \emptyset$). With this latter definition we get a counterpart to Fact 6.1:

FACT 6.10 *If s is a K-atomic path, then for $k^\kappa \in s$ either:*

(a) k is an atomic position; or

(b) k is of γ-type, k_1 its child, and for all j, $1 \leq j \leq \mu_Q(k_1)$, $k_1^{\kappa j} \in S(s)$; or

(c) k is of ν-type, k_1 its child, and for all j, $1 \leq j \leq \mu_M(k_1)$, either

- *$k_1^{\kappa j} \in S(s)$; or*
- *for some $v \in \sigma(pre(k_1^{\kappa j}))$, $v \notin S(s)$.*

The following lemma summarises the relationship between K-atomic paths and paths obtainable by reduction from them.

LEMMA 6.11 *Let s be a K-atomic path. If $u \in D(s)$, there is some ν_0-type position $u' \in D(s)$, whose parent is an element of s such that:*

1. *$u' \trianglelefteq u$.*
2. *For some $v \in \sigma(pre(u'))$, $v \notin S(s)$.*

PROOF. Immediate from the previous fact and the fact that $u \in D(s)$ implies $u \notin S(s)$. ■

A position $u \in D_K(s)$ is said to be *unrestricted for* a path s just in case there is no $\lhd$-greater element in $S(s)$; *i.e.*, no $v \in S(s)$ such that $u \lhd v$. A path s is said to be $\lhd$-*compatible* just in case:

1. Every position of $D_K(s)$ is unrestricted for s.
2. For all $u \in S(s)$, $v \in \sigma(\text{pre}(u))$ implies $v \in S(s)$.

The first clause is identical to that given in the definition of $\lhd$-compatibility for idealisable logics, except with $D_K(s)$ in place of $D(s)$. The second condition is an addition.

We say that a non-atomic, $\lhd$-compatible path, s, is *properly reducible* on a position $u \in s$ just in case the children of u introduced by the reduction are $\lhd$-least elements of $\mathcal{D}_K(s)$. Under these definitions we have:

LEMMA 6.12 *The root path $\{k_0\}$ is $\lhd$-compatible.*

PROOF. Notice that $S(\{k_0\}) = \{k_0\}$, and that k_0 is the $\lhd$-least position.

■

We now come to the main lemma for the non-idealisable logics.

LEMMA 6.13 *If s is a $\lhd$-compatible path which is not K-atomic, there is some element $u \in s$ on which s is properly reducible. Moreover, the paths that result from such a proper reduction are themselves $\lhd$-compatible.*

PROOF. The proof is similar to the proof of Lemma 6.8 above. Since s is not K-atomic, $\mathcal{D}_K(s)$ is non-empty. The irreflexivity of $\lhd$ ensures the existence of at least one $\lhd$-least element of $\mathcal{D}_K(s)$. Likewise we can show the existence of an element of s, all of whose children are $\lhd$-least in $\mathcal{D}_K(s)$. s is thus properly reducible on this position.

Let u be the appropriate element of s. Consider a path obtained from the (proper) reduction of s on u. In order to show $\lhd$-compatibility of s', we must show that:

1. every position of $\mathcal{D}_K(s')$ is unrestricted for s', and
2. for all $u \in S(s')$, $v \in \sigma(\mathrm{pre}(u))$ implies $v \in S(s')$.

The proof of the first condition is identical to that given for the idealisable logics in Lemma 6.8.

To prove the second condition, we notice that only π and ν reductions introduce positions whose prefixes (potentially) differ from their parents. Let u_0 be the child of u introduced by the reduction. In both cases $S(s') = S(s) \cup \{u_0\}$ and $\mathrm{pre}(u_0) = \mathrm{pre}(u)\, u_0$. Consequently:

$$\sigma(\mathrm{pre}(u)) = \sigma(\mathrm{pre}(u))\, \sigma(u_0)\,.$$

By hypothesis (s $\lhd$-compatible) the elements of $\sigma(\mathrm{pre}(u))$ are already elements of $S(s)$, and hence $S(s')$. We are left to show that for all $v \in \sigma(u_0)$, $v \in S(s')$.

1. π-reductions: Notice that $\sigma(u_0) = u_0$ (u_0 is of π_0 type) and $u_0 \in S(s')$.
2. ν-reductions: Suppose for some $v \in \sigma(u_0)$, $v \notin S(s')$. We derive a contradiction. Since $u_0 \in D_K(s)$, $v \in S(s) \cup D_K(s)$ (Fact 6.9). Therefore, $v \in D_K(s)$ ($S(s) \subseteq S(s')$). But, $v \in \sigma(u_0)$ implies $v \lhd u_0$, which contradicts, u_0 being a $\lhd$-least element of $D_K(s)$.

Therefore, in both cases, s' is $\lhd$-compatible. ■

6.3.2.3 Reduction to atomic paths.

Lemmas 6.7 and 6.8 for the idealisable logics, and Lemmas 6.12 and 6.13 for the non-idealisable logics, indicate that we can successfully reduce the $\lhd$-compatible root path to, in the case of the idealisable logics, the set of atomic paths through X^μ, and in the case of the non-idealisable logics, the set of $\lhd$-compatible K-atomic paths, while maintaining the $\lhd$-compatibility of every intermediate path considered.

PROPOSITION 6.14 *Starting from the singleton set consisting of the root path through X^μ, there exists a sequence of proper path reductions by which the atomic paths (K-atomic paths) through X^μ may be enumerated. That is, if k_0 is the root position of X^μ, the procedure:*

$W := \{k_0\}$;

While there is a non-atomic path (non K-atomic path) s in W do:

1. *Properly reduce s;*
2. *Replace s in W by the paths that result from this reduction;*

terminates, resulting in the set of atomic paths (K-atomic paths) through X^μ. Furthermore, at every stage, all the elements of W are $\lhd$-compatible.

PROOF. Lemma 6.7 (Lemma 6.12) ensures that the initial element of W is $\lhd$-compatible, while Lemma 6.8 (Lemma 6.13) shows that this property is an invariant for the loop for the idealisable (non-idealisable) logics. Termination can be established from the fact that X^μ contains finitely many positions, and a path, s', that results from the reduction of s, satisfies: $D(s') \subset D(s)$ $(D_K(s') \subset D_K(s))$. A well-ordering which decreases at each iteration is easy to construct. ■

6.3.2.4 K-atomic paths.

For the idealisable logics, Proposition 6.14 guarantees the proper reduction of the root path to the atomic paths through X^μ. The existence of a set of connections that span X^μ ensures that each atomic path contains a σ-complementary connection. This is the desired situation as outlined in the introduction to this section. In the following sections we justify proper reductions.

For the non-idealisable logics, however, we can only reduce as far as the K-atomic paths using proper reductions. Recall that for the non-idealisable logics a path t, containing a σ-complementary connection $\{ u_1, u_2 \}$, is only σ-complementary itself if for every $v \in \sigma(\mathrm{pre}(u))$, $v \in S(t)$. We now show that the existence of a set of σ-complementary connections that makes every atomic path through X^μ σ-complementary in this sense, entails that every K-atomic path through X^μ is also σ-complementary.

In what follows we understand the term: "$\lhd$-compatible," and the term: "σ-complementary," in the sense of the non-idealisable logics. We need the following lemmata:

LEMMA 6.15 *Let s be a $\lhd$-compatible K-atomic path. Furthermore, let $\{ u_1, u_2 \}$ be a σ-complementary connection. If $u_i \in s$, $i = 1, 2$, s is σ-complementary.*

PROOF. Since the connection is σ-complementary we have $\sigma(\mathrm{pre}(u_1)) = \sigma(\mathrm{pre}(u_2))$. We must show that for all $v \in \sigma(\mathrm{pre}(u_1))$, $v \in S(s)$. But this

follows immediately from the definition of $\lhd$-compatibility (for the non-idealisable logics). ■

The main proposition is as follows:

PROPOSITION 6.16 *If U is a set of σ-complementary connections that span X^{μ}, and s is a $\lhd$-compatible K-atomic path, s is σ-complementary.*

PROOF. If s is an atomic path the result follows by hypothesis and Lemma 6.15. Suppose then that s is not an atomic path. Let t be an atomic path reachable by reductions (possibly not proper) from s. By hypothesis there is a σ-complementary connection $\{ u_1, u_2 \} \in U$ that makes t σ-complementary. Therefore, for all $v \in \sigma(\mathrm{pre}(u_i))$, $i = 1,2$, we have $v \in \mathcal{S}(t)$, by definition. Furthermore, for any such v, we must have $v \in \mathcal{S}(s)$ or $v \in \mathcal{D}(s)$ (since t is obtained from s by reductions).

If $u_i \in \mathcal{S}(s)$ then $u_i \in s$, $i = 1,2$, since the u_i are atomic positions. By Lemma 6.15, s is σ-complementary.

Therefore, without loss of generality, suppose that $u_1 \notin s$, *i.e.*, $u_1 \notin \mathcal{S}(s)$. We derive a contradiction.

Since $u_1 \in t$ but $u_1 \notin s$, we have: $u_1 \in \mathcal{D}(s)$ (t is obtained from s by reductions). Let u' be the ν_0-type position, with $u' \trianglelefteq u_1$, whose parent is an element of s. Such an u' must exist by Lemma 6.11. Since $u' \trianglelefteq u_1$, $\mathrm{pre}(u') \preceq \mathrm{pre}(u_1)$, and consequently,

$$\sigma(\mathrm{pre}(u')) \preceq \sigma(\mathrm{pre}(u_1)) .$$

Since $u' \notin \mathcal{D}_K(s)$ (s is K-atomic), there must be a $v' \in \sigma(\mathrm{pre}(u'))$ such that $v' \notin \mathcal{S}(s)$ (nor of $\mathcal{D}_K(s)$ for that matter). Since $v \in \sigma(\mathrm{pre}(u'))$, $v \in \sigma(\mathrm{pre}(u_1))$ and hence must be an element of $\mathcal{D}(s)$. By Condition B on modal substitutions we have:

$$\sigma(\mathrm{pre}(v')) \preceq \sigma(\mathrm{pre}(u')) .$$

Let u'' be the ν_0-type position with $u'' \ll v'$, whose parent is an element of s (Lemma 6.11). $u'' \ll v'$ implies $u'' \lhd v'$. Since $v' \in \sigma(\mathrm{pre}(u'))$, we have $v' \lhd u'$, and hence $u'' \lhd u'$, by the transitivity of $\lhd$. u'' must be distinct from u' since $\lhd$ is irreflexive. Moreover, since $u'' \ll v'$, $\mathrm{pre}(u'') \preceq \mathrm{pre}(v')$, and

$$\sigma(\mathrm{pre}(u'')) \preceq \sigma(\mathrm{pre}(v')) .$$

Therefore:

$$\sigma(\mathrm{pre}(u'')) \preceq \sigma(\mathrm{pre}(v')) \preceq \sigma(\mathrm{pre}(u')) \preceq \sigma(\mathrm{pre}(u_1)) .$$

Thus, for all $v \in \sigma(\mathrm{pre}(u''))$, $v \in S(t)$. We can repeat the reasoning to obtain a u''', u'''' *etc*, all elements of $D(s)$, such that:

$$\cdots \lhd u'''' \lhd u''' \lhd u'' \lhd u' \lhd u_1.$$

Suppose $u*$ is the $\lhd$-least of this sequence (which must exist since there are only a finite number of positions and $\lhd$ is irreflexive). We have that for all $v \in \sigma(\mathrm{pre}(u*))$, $v \in S(t)$; and consequently, for such a v, either $v \in S(s)$ or $v \in D(s)$ as before. By assumption, $v \notin D(s)$, else $u*$ could not be $\lhd$-least in $D(s)$. Hence $v \in S(s)$. But then $u* \in D_K(s)$ and s is not K-atomic, contradicting the hypothesis. ■

6.3.3 Correctness of proper reductions.

In the previous section we showed that the root path could be properly reduced to the set of atomic, or K-atomic paths through X^μ. Moreover, if X^μ is spanned, by a set of σ-complementary connections, every atomic or K-atomic path through X^μ is σ-complementary(for the appropriate definition of "σ-complementary" for paths.)

Roughly speaking, we consider a path to be *$\mathcal{L}$-satisfiable* if the set of positions associated with it can be interpreted in an $\mathcal{L}$-model in a natural way. In this section we show that proper reductions preserve $\mathcal{L}$-satisfiability, and that σ-complementary paths are not $\mathcal{L}$-satisfiable.

Recall that we use $P(\mu)$ to denote the set of prefixes in X^{μ}. We need to specialise this notation as follows: let $P(\mu, s)$ denote the set of prefixes of the positions in $S(s)$, *i.e.*,

$$P(\mu, s) \stackrel{\text{df}}{=} \{ \text{pre}(u) \mid u \in S(s) \},$$

and $P_\sigma(\mu, s)$ the "image" of $P(\mu, s)$ under σ, *i.e.*,

$$P_\sigma(\mu, s) \stackrel{\text{df}}{=} \{ p \mid p \preceq \sigma(q)\,,\ q \in P(\mu, s) \}.$$

Notice that if q is the image of the prefix of some element of $S(s)$ under σ, then $P_\sigma(\mu, s)$ contains all initial subsequences of q. It is the set of prefixes that we shall interpret into a model.

We need similar notions for constants. By the definition of the label of a position, the elements of $T_Q(\mu)$ that can possibly appear in the label of an indexed position must dominate that position in the formula tree. We use $C(\mu, s)$ to denote this set thus:

$$C(\mu, s) \stackrel{\text{df}}{=} T_Q(\mu) \cap S(s)$$

Let $C_\sigma(\mu, s)$ denote the image of $C(\mu, s)$ under σ, together with the individual constants of A, *i.e.*,

$$C_\sigma(\mu, s) \stackrel{\text{df}}{=} \{ \sigma(u) \mid u \in C(\mu, s) \} \cup \{\text{constants of } A\}.$$

These definitions are the quantifier counterparts to the definitions of $P(\mu, s)$ and $P_\sigma(\mu, s)$ above. $C_\sigma(\mu, s)$ is the set of constants that we shall interpret into a model.

Since the notation will get cumbersome otherwise, we shall usually omit mention of μ. That is, we use $P(s)$ in place of $P(\mu, s)$, $P_\sigma(s)$ instead of $P_\sigma(\mu, s)$, $C(s)$ instead of $C(\mu, s)$ and $C_\sigma(s)$ instead of $C_\sigma(\mu, s)$.

For a set of prefixes P we use $|P|$ to denote the set of positions that comprise the prefixes:

$$|P| \stackrel{\text{df}}{=} \{ u \in p \mid p \in P \}.$$

The following facts are immediate consequences of these definitions and Fact 6.3:

FACT 6.17 $|P(s)| \subseteq S(s)$.

FACT 6.18 $C(s) \subseteq S(s)$.

Let $\langle G, R, D, \bar{D}, \Vdash \rangle$ be an $\mathcal{L}$-model and $\langle G_0, R_0 \rangle$ be an $\mathcal{L}$-frame with $G_0 \cap G = \emptyset$. Let p, q denote elements of G_0. (G_0 will be our set of prefixes and R_0 the accessibility relation on prefixes.) Let D_0 be a set of constants distinct from D. A mapping:

$$\iota: P_0 \subseteq G_0 \mapsto G \quad \text{and} \quad \iota: C_0 \subseteq D_0 \mapsto D$$

is an *$\mathcal{L}$-interpretation* for P_0 and C_0 in $\langle G, R, D, \bar{D}, \Vdash \rangle$ if, for any $p, q \in P_0$,

$$p \; R_0 \; q \quad \text{implies} \quad \iota(p) \; R \; \iota(q) \, .$$

We say that a path, s, is *$\mathcal{L}$-satisfiable under σ* just in case there is an $\mathcal{L}$-model, $\langle G, R, D, \bar{D}, \Vdash \rangle$, and an $\mathcal{L}$-interpretation, ι, of $P_\sigma(s)$ into G and $C_\sigma(s)$ into D such that:

1. For each $u \in S(s)$, with $p = \sigma(\text{pre}(u))$ and $Y = \sigma(\text{sform}(u))$, all the constants of $\iota(Y)$ are in $\bar{D}(\iota(p))$ and:

$$\iota(p) \Vdash \iota(Y) \, .$$

2. For each $v \in C_\sigma(s)$ with $q = \sigma(\text{pre}(v))$:

$$q \in P_\sigma(s) \quad \text{implies} \quad \iota(v) \in \bar{D}(\iota(q)).$$

The first condition is the formal expression of the informal argument of the introduction. The prefix, $\text{pre}(u)$, of a position, u, is used as the name of a point at which the signed formula, $\text{sform}(u) = \langle \text{lab}(u), \text{pol}(u) \rangle$, associated with the position is forced. We require this relationship to hold under a mapping, σ, of "variable" elements of the prefix, and "variables" in the atoms of the formula. "Variables" represent a degree of choice in the point or parameter named.

The second condition, called the *Parameter Condition*, ensures that parameters are interpreted in a consistent way. The condition ensures that the constant

of the model denoted by v is an element of the domain of the point of the model denoted by the prefix of v. This condition is only of interest for the varying and cumulative domain logics as the following lemma shows.

LEMMA 6.19 *Let $\mathcal{L}$ denote a constant domain logic under consideration. If the first condition for $\mathcal{L}$-satisfiability under σ holds for a path s, so does the Parameter Condition.*

PROOF. Suppose $v \in C_\sigma(s)$ with $q = \sigma(\mathrm{pre}(v))$, and $q \in P_\sigma(s)$. The first condition on $\mathcal{L}$-satisfiability gives us a model, $\langle G, R, D, \bar{D}, \Vdash \rangle$, and an interpretation, ι, of $P_\sigma(s)$ and $C_\sigma(s)$ in that model. Thus $\iota(v) \in \bar{D}(w)$, for some $w \in G$, and $\iota(q) = w'$, for some $w' \in G$. But $\bar{D}(w) = \bar{D}(w')$, since $\mathcal{L}$ has constant domains. Hence $\iota(v) \in \bar{D}(\iota(q))$. ■

The Parameter Condition plays a crucial rôle only in the justification of proper γ reductions (Proposition 6.27) for these logics. We shall remark on it there. The condition must be checked for each reduction however. To that end, we simplify the other proofs by means of the following lemmata.

For application in the case of α and β reductions:

LEMMA 6.20 *Let s be a $\lhd$-compatible path through X^μ, $\mathcal{L}$-satisfiable under σ. Let s' be a path resulting from the proper reduction of s on a position $u \in s$. If $P_\sigma(s') = P_\sigma(s)$ and $C_\sigma(s') = C_\sigma(s)$, then s' satisfies the Parameter Condition.*

PROOF. Since $C_\sigma(s') = C_\sigma(s)$ and $P_\sigma(s') = P_\sigma(s)$ the Parameter Condition must hold for s' since it held for s by hypothesis. ■

For application in the case of π and ν reductions:

LEMMA 6.21 *Let $\mathcal{L}$ denote*

1. *a varying domain logic; or*
2. *a cumulative domain variant of one of K, K4, D, D4, T, S4.*

Let s be a $\lhd$-compatible path through X^μ. For $p \in T_M(\mu)^*$, if $p \notin P_\sigma(s)$, there is no $v \in C_\sigma(s)$ with $\sigma(pre(v)) = p$.

PROOF. Suppose not. That is suppose for $v \in C_\sigma(s)$, $\sigma(\text{pre}(v)) = p$ and $p \notin P_\sigma(s)$. First notice that any such v cannot have $v \in C(s)$, because in that case $v \in S(s)$ (since $C(s) \subseteq S(s)$) and by definition: $p \in P_\sigma(s)$, which contradicts $p \notin P_\sigma(s)$.

So there must be a $u \in C(s)$ with $\sigma(u) = v$. Suppose $\sigma(\text{pre}(u)) = q$. By the reasoning above, $q \in P_\sigma(s)$.

1. For varying domains: since σ is $\mathcal{L}$-admissible, we have: $q = p$, *i.e.*, $p \in P_\sigma(s)$, which contradicts $p \notin P_\sigma(s)$.
2. For cumulative domain variants of K, K4, D, D4, T, S4: since σ is $\mathcal{L}$-admissible, we have either: (a) q =p; or (b) $p \, R_0 \, q$.

   The first case we dealt with above. For these logics, Fact 6.4 gives us that $p \preceq q$. But then, $p \in P_\sigma(s)$, by definition of $P_\sigma(s)$, which again contradicts $p \notin P_\sigma(s)$.

■

IMPORTANT NOTATIONAL POINT. In what follows we shall abuse our notation and use principal and secondary types to denote arbitrary positions of that type. In particular, if we say: "a path s is properly reducible on π," we mean that there is a position $u \in s$ such that $\text{Ptype}(u) = \pi$ and s is properly reducible on u. Furthermore, in this context we shall use π_0 to denote the child of u. Similar abuses are extended to the other types. (END OF POINT.)

6.3.3.1 Complementary paths.

PROPOSITION 6.22 *A path s through X^μ which contains a σ-complementary connection $\{ u, v \}$ is not $\mathcal{L}$-satisfiable.*

PROOF. By definition of a σ-complementary connection:

- $\sigma(\text{pre}(u)) = p = \sigma(\text{pre}(v))$, for some prefix $p \in T_M(\mu)^*$.
- $\sigma(\text{lab}(u)) = P = \sigma(\text{lab}(v))$, for some atomic formula P.
- $\text{pol}(u) \neq \text{pol}(v)$.

Suppose, for a contradiction, that s is $\mathcal{L}$-satisfiable with $\langle G, R, D, \bar{D}, \Vdash \rangle$ and ι the appropriate $\mathcal{L}$-model and $\mathcal{L}$-interpretation. Then, by definition, all the constants of $\iota(P)$ are in $\bar{D}(\iota(p))$, and:

$$\iota(p) \Vdash \iota(\sigma(\text{sform}(u))) \quad \text{and} \quad \iota(p) \Vdash \iota(\sigma(\text{sform}(v))),$$

i.e.,

$$\iota(p) \Vdash \iota(\langle P, 0 \rangle) \quad \text{and} \quad \iota(p) \Vdash \iota(\langle P, 1 \rangle),$$

or equivalently:

$$\iota(p) \nVdash \iota(P) \quad \text{and} \quad \iota(p) \Vdash \iota(P),$$

since the polarities of the positions are different. This contradicts condition (1) Corollary 4.1 for models. Hence s is not $\mathcal{L}$-satisfiable under σ. ■

REMARK. Notice that we extend ι and σ, to signed formulae thus: for a position u,

$$\iota(\langle \text{lab}(u), \text{pol}(u) \rangle) = \langle \iota(\text{lab}(u)), \text{pol}(u) \rangle$$
$$\sigma(\langle \text{lab}(u), \text{pol}(u) \rangle) = \langle \sigma(\text{lab}(u)), \text{pol}(u) \rangle.$$

We shall do this extensively in the sequel without remark. Furthermore, both ι and σ leave the structure of formulae unchanged, mapping only the arguments of predicate symbols. Consequently the mappings preserve the types of signed formulae. (END OF REMARK.)

6.3.3.2 Propositional reductions.

We justify proper α and β reductions.

PROPOSITION 6.23 *Let s be a $\lhd$-compatible path through X^μ, $\mathcal{L}$-satisfiable under σ, and properly reducible on α. Then the path obtained by reduction on α is $\mathcal{L}$-satisfiable under σ.*

PROOF. The path obtained by reduction of s on α is $(s \setminus \{\alpha\}), \alpha_1, \alpha_2$. Call this path s'. We have:

- $S(s') = S(s) \cup \{\alpha_1, \alpha_2\}$.
- $P_\sigma(s') = P_\sigma(s)$.
- $C_\sigma(s') = C_\sigma(s)$.

Let p be the image of the prefix of α under σ. Then p is also the image of the prefixes of the α_i, $i = 1, 2$, under σ. Let Y be the image of $\mathrm{sform}(\alpha)$ under σ. Note that Y is a signed formula of α type. We shall use Y_1 and Y_2 to denote its immediate subformulae. Under these conventions:

$$Y_1 = \sigma(\mathrm{sform}(\alpha_1)) \quad \text{and} \quad Y_2 = \sigma(\mathrm{sform}(\alpha_2))$$

by the definition of labels. By hypothesis we have the existence of an $\mathcal{L}$-model, $\langle G, R, D, \tilde{D}, \Vdash \rangle$, and an $\mathcal{L}$-interpretation, ι, for which:

$$\iota(p) \Vdash \iota(Y)\,.$$

Hence, by the model conditions for α-type formulae (Corollary 4.1):

$$\iota(p) \Vdash \iota(Y_1) \quad \text{and} \quad \iota(p) \Vdash \iota(Y_2)\,;$$

that is:

$$\iota(p) \Vdash \iota(\sigma(\mathrm{sform}(\alpha_1))) \quad \text{and} \quad \iota(p) \Vdash \iota(\sigma(\mathrm{sform}(\alpha_2)))\,.$$

The Parameter Condition is satisfied by s' by Lemma 6.20. Consequently, the reduced path is $\mathcal{L}$-satisfiable under σ. ■

PROPOSITION 6.24 *Let s be a $\lhd$-compatible path through X^μ, $\mathcal{L}$-satisfiable under σ, and properly reducible on β. At least one of the paths obtained by reduction on β is $\mathcal{L}$-satisfiable under σ.*

PROOF. Two paths result from the reduction of s on β: $(s \setminus \{\beta\}), \beta_1$ and $(s \setminus \{\beta\}), \beta_2$. Call these s_1 and s_2 respectively. We have: for $i = 1, 2$,

- $S(s_i) = S(s) \cup \{\beta_i\}$.
- $P_\sigma(s_i) = P_\sigma(s)$.
- $C_\sigma(s_i) = C_\sigma(s)$.

If p is the image of the prefix of β under σ, then p is also the prefix of β_1 and β_2. Let Y be the image of sform(β) under σ. Note that Y is a signed formula of β type. We shall use Y_1 and Y_2 to denote its immediate subformulae. Under these conventions:

$$Y_1 = \sigma(\text{sform}(\beta_1)) \quad \text{and} \quad Y_2 = \sigma(\text{sform}(\beta_2))$$

by the definition of labels. By hypothesis we have the existence of an $\mathcal{L}$-model, $\langle G, R, D, \tilde{D}, \Vdash \rangle$, and an $\mathcal{L}$-interpretation, ι, for which:

$$\iota(p) \Vdash \iota(Y).$$

By the model conditions for β-type formulae (Corollary 4.1), either:

$$\iota(p) \Vdash \iota(Y_1) \quad \text{or} \quad \iota(p) \Vdash \iota(Y_2);$$

That is:

$$\iota(p) \Vdash \iota(\sigma(\text{sform}(\beta_1))) \quad \text{or} \quad \iota(p) \Vdash \iota(\sigma(\text{sform}(\beta_2))).$$

The Parameter Condition is satisfied by s_i, $i = 1, 2$, by Lemma 6.20. Consequently, at least one of the resulting paths is $\mathcal{L}$-satisfiable under σ. ■

6.3.3.3 Quantifier reductions.

Before proving the correctness of proper δ reductions, we prove an auxilliary lemma, the force of which is that the "parameters" (elements of Δ_0) introduced by proper δ reductions are indeed new to the "sequent." This follows from the $\lhd$-compatibility of the path.

LEMMA 6.25 *Let s be a $\lhd$-compatible path, reducible on δ, then $\delta_0 \notin C_\sigma(s)$.*

PROOF. Suppose, for a contradiction, that $\delta_0 \in C_\sigma(s)$. Since $\delta_0 \in D(s)$, $\delta_0 \notin S(s)$, and hence $\delta_0 \notin C(s)$ (by Fact 6.18). Thus there must be some $v \in C(s)$ with $\sigma(v) = \delta_0$, *i.e.*, $\delta_0 \lhd v$. But there can be no such $v \in C(s) \subseteq S(s)$ since δ_0 is unrestricted for s. ■

We now justify proper δ reductions.

PROPOSITION 6.26 *Let s be a $\lhd$-compatible path through X^μ, $\mathcal{L}$-satisfiable under σ, and properly reducible on δ. The path obtained by reduction on δ is $\mathcal{L}$-satisfiable under σ.*

PROOF. The path obtained by reduction on δ is $(s \setminus \{\delta\}), \delta_0$. Call this path s'. We have:

- $S(s') = S(s) \cup \{\delta_0\}$.
- $P_\sigma(s') = P_\sigma(s)$.
- $C_\sigma(s') = C_\sigma(s) \cup \{\delta_0\}$.

Let p be the image of the prefix of δ (and hence of δ_0) under σ. Let Y be the image of $\mathrm{sform}(\delta)$ under σ. Note that Y is a signed formula of δ type. We shall use Y_0 to denote its immediate subformula with the individual quantified variable free, and $Y_0(a)$, for some a, to indicate substitution of a for that free variable. The particular a we are interested in, of course, is the position δ_0 itself. Under these conventions:

$$Y_0(\delta_0) = \sigma(\mathrm{sform}(\delta_0))$$

by the definition of labels and since $\sigma(\delta_0) = \delta_0$. By hypothesis we have an $\mathcal{L}$-model, $\langle G, R, D, \bar{D}, \Vdash \rangle$, and an $\mathcal{L}$-interpretation, ι, for which: all constants of $\iota(Y)$ are in $\bar{D}(\iota(p))$, and:

$$\iota(p) \Vdash \iota(Y)\,.$$

Hence, by the model conditions (Corollary 4.1), there is some $c \in \bar{D}(\iota(p))$ such that:

$$\iota(p) \Vdash \iota(Y_0)(c).$$

(Remember, ι and σ map only constants. $\iota(Y_0)$ is therefore a δ formula over the language of the model with a free individual variable. We use $\iota(Y_0)(c)$ to denote the substitution of c, a constant of the model, for that variable.) Since $\delta_0 \notin C_\sigma(s)$ (Lemma 6.25), ι is undefined for δ_0. We extend ι to $C_\sigma(s')$ as follows:

$$\iota'(u) = \begin{cases} c, & u = \delta_0 \\ \iota(u), & u \in C_\sigma(s). \end{cases}$$

The extended mapping is an $\mathcal{L}$-interpretation for $S(s')$ since δ_0 is the only additional constant, and ι was an $\mathcal{L}$-interpretation for $S(s)$. Since $\iota'(\delta_0) = c$ and $\iota'(p) = \iota(p)$, we have: $\iota'(\delta_0) \in \bar{D}(\iota'(p))$. Now, since:

$$\iota(p) \Vdash \iota(Y_0)(c),$$

and ι and ι' agree on the constants of Y_0 and p:

$$\iota'(p) \Vdash \iota'(Y_0)(c).$$

Furthermore:

$$\iota'(Y_0)(c) = \iota'(Y_0)(\iota'(\delta_0)) = \iota'(Y_0(\delta_0)) = \iota'(\sigma(\mathrm{sform}(\delta_0))).$$

Consequently:

$$\iota'(p) \Vdash \iota'(\sigma(\mathrm{sform}(\delta_0))).$$

For $u \in S(s)$:

$$\iota(\sigma(\mathrm{pre}(u))) \Vdash \iota(\sigma(\mathrm{sform}(u)))$$

by hypothesis, and since ι and ι' agree on $C_\sigma(s)$, we have:

$$\iota'(\sigma(\mathrm{pre}(u))) \Vdash \iota'(\sigma(\mathrm{sform}(u))).$$

The Parameter Condition is satisfied for $C_\sigma(s')$ since it was satisfied for $C_\sigma(s)$ by hypothesis, and for δ_0 we have: $\sigma(\mathrm{pre}(\delta_0)) = p$, $p \in P_\sigma(s')$ and $\iota'(\delta_0) \in \bar{D}(\iota'(p))$ by construction. s' is therefore $\mathcal{L}$-satisfiable under σ. ■

PROPOSITION 6.27 *Let s be a $\lhd$-compatible path through X^μ, $\mathcal{L}$-satisfiable under σ, and properly reducible on γ. The path obtained by reduction on γ is $\mathcal{L}$-satisfiable under σ.*

PROOF. Let k^κ be the γ-type position in question and l the child of k in the (unindexed) formula tree. By definition, the secondary type of l is γ_0. k^κ has $\mu(l)$ children in the indexed formula tree X^μ. For s to be reducible on k^κ it means that there is at least one j, $1 \leq j \leq \mu(l)$ such that $l^{\kappa j} \in \mathcal{D}(s)$ with $l^{\kappa j}$ being a $\lhd$-least element of $\mathcal{D}(s)$. Let γ_0 denote the position $l^{\kappa j}$ for some such j. We continue as in the δ case above.

The path obtained by reduction on γ is s, γ_0. Call this path s'. We have:

- $S(s') = S(s) \cup \{\gamma_0\}$.
- $P_\sigma(s') = P_\sigma(s)$.
- $C_\sigma(s') = C(s) \cup \{\gamma_0\}$.

Let p be the image of the prefix of γ (and hence of γ_0) under σ. Let Y be the image of sform(γ) under σ. Note that Y is a signed formula of γ type. We shall use Y_0 to denote its immediate subformula with the individual quantified variable free, and $Y_0(a)$, for some a, to indicate substitution of a for that free variable. The particular a we are interested in, of course, is the image, v, of γ_0 under σ. Under these conventions:

$$Y_0(v) = \sigma(\text{sform}(\gamma_0))\,,$$

by the definition of labels, and since $\sigma(\gamma_0) = v$. By hypothesis we have an $\mathcal{L}$-model, $\langle G, R, D, \bar{D}, \Vdash \rangle$, and an $\mathcal{L}$-interpretation, ι, for which: all constants of $\iota(Y)$ are in $\bar{D}(\iota(p))$, and:

$$\iota(p) \Vdash \iota(Y)\,.$$

Hence, by the model conditions (Corollary 4.1), for every $c \in \bar{D}(\iota(p))$:

$$\iota(p) \Vdash \iota(Y_0)\,(c).$$

Let $\sigma(\text{pre}(v)) = q$. Since v is a constant of $Y_0(v)$, we need only show how to extend ι to ι', an $\mathcal{L}$-interpretation for $C_\sigma(s')$, such that:

- $\iota'(v) \in \bar{D}(\iota'(p))$, and,
- the Parameter Condition holds for s'.

We consider each domain condition in turn.

Constant domains: This is the simplest case. The Parameter Condition is satisfied for any extension of ι by Lemma 6.19.

1. Suppose $v \in C_\sigma(s)$. Then $C_\sigma(s) = C_\sigma(s')$. ι is already defined on v and $\iota(v) \in \bar{D}(w)$ for some $w \in G$. But $\bar{D}(w) = \bar{D}(\iota(p))$ by assumption (constant domains). Define: $\iota' = \iota$.
2. Suppose $v \notin C_\sigma(s)$. Choose $c \in \bar{D}(\iota(p))$ (one exists since every domain of the model is non-empty). Extend ι as follows:
$$\iota'(u) = \begin{cases} c, & u = v \\ \iota(u), & u \in C_\sigma(s). \end{cases}$$
Clearly ι' is an $\mathcal{L}$-interpretation for $C_\sigma(s')$, and $\iota'(v) \in \bar{D}(\iota'(p))$ by construction.

Varying domains: The extra condition on varying domain admissibility states that: for $u', v' \in T_Q(\mu)$, if $\sigma(u') = v'$, then $\sigma(\text{pre}(u')) = \sigma(\text{pre}(v'))$.

Since: $\sigma(\gamma_0) = v$, $p = q$ under this condition; *i.e.*, $\bar{D}(\iota(p)) = \bar{D}(\iota(q))$. Extend ι as in the constant domain case depending on whether $v \in C_\sigma(s)$ or not.

1. Suppose $v \in C_\sigma(s)$. Then $C_\sigma(s) = C_\sigma(s')$. ι is already defined on v and $\iota(v) \in \bar{D}(\iota(q))$ by the Parameter Condition for s. Hence: $\iota(v) \in \bar{D}(\iota(p))$, since $\bar{D}(\iota(p)) = \bar{D}(\iota(q))$. Define: $\iota' = \iota$. The Parameter Condition holds for s' by Lemma 6.20.
2. Suppose $v \notin C_\sigma(s)$. Choose $c \in \bar{D}(\iota(p))$. (All domains are non-empty.) Extend ι as follows:
$$\iota'(u) = \begin{cases} c, & u = v \\ \iota(u), & u \in C_\sigma(s). \end{cases}$$
Clearly ι' is an $\mathcal{L}$-interpretation for $C_\sigma(s')$. $\iota(v) \in \bar{D}(\iota(p))$ by construction, and the Parameter Condition holds for s', since $\bar{D}(\iota(p)) = \bar{D}(\iota(q))$.

Cumulative domains: The extra condition on cumulative domain admissibility is: for $u', v' \in T_Q(\mu)$, if $\sigma(u') = v'$, then either: (a) $\sigma(\text{pre}(u')) = \sigma(\text{pre}(v'))$; or (b) $\sigma(\text{pre}(v'))\ R_0\ \sigma(\text{pre}(u'))$. Since $\sigma(\gamma_0) = v$, we have either: (a) $p = q$, or (b) $q\ R_0\ p$.

(a) $p = q$. Then $\bar{D}(\iota(p)) = \bar{D}(\iota(q))$. Proceed as in the varying domain case above.

(b) $q\ R_0\ p$. $q\ R_0\ p$ implies $q \preceq p$ for the logics K, K4, D, D4, T, S4 by Fact 6.4. Consequently $q \in \mathcal{P}_\sigma(s)$, by definition of $\mathcal{P}_\sigma(s)$. Since ι is an $\mathcal{L}$-interpretation for $\mathcal{P}_\sigma(s)$ we have: $\iota(q)\ R\ \iota(p)$, and thus, since the model has cumulative domains, $\bar{D}(\iota(q)) \subseteq \bar{D}(\iota(p))$.

i. Suppose $v \in \mathcal{C}_\sigma(s)$. Then $\mathcal{C}_\sigma(s') = \mathcal{C}_\sigma(s)$. Since $q \in \mathcal{P}_\sigma(s)$ and the Parameter Condition holds for s, $\iota(v) \in \bar{D}(\iota(q))$. Consequently: $\iota(v) \in \bar{D}(\iota(p))$, since $\bar{D}(\iota(q)) \subseteq \bar{D}(\iota(p))$. The Parameter Condition holds for s' by Lemma 6.20.

ii. Suppose $v \notin \mathcal{C}_\sigma(s)$. Choose $c \in \bar{D}(\iota(q))$. (All domains are non-empty.) Extend ι as follows:

$$\iota'(u) = \begin{cases} c, & u = v \\ \iota(u), & u \in \mathcal{C}_\sigma(s). \end{cases}$$

Clearly ι' is an $\mathcal{L}$-interpretation for $\mathcal{C}_\sigma(s')$. Moreover, $\iota'(v) \in \bar{D}(\iota'(p))$ by construction, and since ι' and ι agree on $\mathcal{P}_\sigma(s)$. The Parameter Condition holds for s' under ι' by the choice of interpretation for v, and the fact that it held for s under ι.

■

6.3.3.4 Modal reductions.

Before proving the correctness of proper π reductions, we prove some auxilliary lemmata, the force of which is that the prefixes introduced by proper π reductions are indeed new to the "sequent." This follows from the $\lhd$-compatibility of the path.

LEMMA 6.28 *Let s be a $\lhd$-compatible path, properly reducible on π, then $\pi_0 \notin |P_\sigma(s)|$.*

PROOF. Suppose $\pi_0 \in |P_\sigma(s)|$. We show that this leads to a contradiction. $\pi_0 \in D(s)$ since s is properly reducible on π. $\pi_0 \in D(s)$ implies $\pi_0 \notin S(s)$, and therefore, $\pi_0 \notin |P(s)|$ (Fact 6.17). Therefore, there must be some $v \in |P(s)|$ with $\pi_0 \in \sigma(v)$, and consequently $\pi_0 \lhd v$. But $v \in |P(s)|$ implies $v \in S(s)$ since $|P(s)| \subseteq S(s)$. Hence, we have $\pi_0 \lhd v$ for some $v \in S(s)$ which contradicts π_0 being unrestricted and hence the $\lhd$-compatibility of s. ■

LEMMA 6.29 *Let s be a $\lhd$-compatible path, properly reducible on π, and $p\pi_0 = \sigma(pre(\pi_0))$. Then*

1. *$p\pi_0 \notin P_\sigma(s)$.*
2. *For the logics K, K4, D, D4, T, S4 only: there is no $q \in P_\sigma(s)$ with $p\pi_0 R_0 q$.*

PROOF.

1. Follows immediately, since $\pi_0 \in p\pi_0$ and $\pi_0 \notin |P_\sigma(s)|$ by the previous lemma.
2. Any such q must have $p\pi_0 \preceq q$ by Fact 6.4. Since $\pi_0 \in p\pi_0$, we must have $\pi_0 \in q$, which contradicts the previous lemma.

■

PROPOSITION 6.30 *Let s be a $\lhd$-compatible path through X^μ, $\mathcal{L}$-satisfiable under σ, and properly reducible on π. The path obtained by reduction on π is $\mathcal{L}$-satisfiable under σ.*

PROOF. The path obtained by reduction on π is $(s \setminus \{\pi\}), \pi_0$. Call this path s'. Let p be the image of the prefix of π under σ; *i.e.*, $p = \sigma(\mathrm{pre}(\pi))$. We have:

- $S(s') = S(s) \cup \{\pi_0\}$.
- $P_\sigma(s') = P_\sigma(s) \cup \{p\pi_0\}$. (S5: $P_\sigma(s') = P_\sigma(s) \cup \{\pi_0\}$.)
- $C_\sigma(s') = C_\sigma(s)$.

Let Y be the image of $\mathrm{sform}(\pi)$ under σ. Note that Y is a signed formula of π type. We shall use Y_0 to denote its immediate subformula; *i.e.*,

$$Y_0 = \sigma(\mathrm{sform}(\pi_0)) .$$

By hypothesis we have the existence of an $\mathcal{L}$-model, $\langle G, R, D, \bar{D}, \Vdash \rangle$, and an $\mathcal{L}$-interpretation, ι, for which:

- All constants of $\iota(Y)$ are in $\bar{D}(\iota(p))$, and: $\iota(p) \Vdash \iota(Y)$.
- For all $v \in C_\sigma(s)$ with $\sigma(\mathrm{pre}(v)) = r$, if $r \in P_\sigma(s)$, $\iota(v) \in \bar{D}(\iota(r))$.

Hence, by the model conditions (Corollary 4.1) there is some point $w \in G$, with $\iota(p)\ R\ w$, such that:

$$w \Vdash \iota(Y_0) .$$

Since $p\pi_0 \notin P_\sigma(s)$ (S5: $\pi_0 \notin P_\sigma(s)$) by Lemma 6.29, ι is undefined for $p\pi_0$. We extend ι to $P_\sigma(s')$ as follows:

$$\iota'(q) = \begin{cases} w, & q = p\pi_0 \quad (\text{S5: } q = \pi_0) \\ \iota(q), & \text{otherwise.} \end{cases}$$

Assuming the extended mapping is an $\mathcal{L}$-interpretation, we have by construction: for $u \in S(s)$,

$$\iota'(\sigma(\mathrm{pre}(u))) \Vdash \iota'(\sigma(\mathrm{sform}(u)))$$

since ι and ι' agree on $P_\sigma(s)$ and $C_\sigma(s)$. In addition, by choice, we have:

$$\iota'(p\pi_0) \Vdash \iota'(Y_0)$$

i.e.,

$$\iota'(p\pi_0) \Vdash \iota'(\sigma(\mathrm{sform}(\pi_0)))$$

since $\iota'(p\pi_0) = w$.

The Parameter Condition is satisfied for s' under ι' for constant domains by Lemma 6.19. For varying and cumulative domains, by Lemma 6.21, $p\pi_0 \notin \mathcal{P}_\sigma(s)$ implies there can be no $v \in \mathcal{C}_\sigma(s)$ with $\sigma(\mathrm{pre}(v)) = p\pi_0$. The Parameter Condition holds in these cases since it held for s under ι, and ι and ι' agree on $\mathcal{P}_\sigma(s)$ and $\mathcal{C}_\sigma(s)$. s' is therefore $\mathcal{L}$-satisfiable under σ.

We have only to show that ι' is indeed an $\mathcal{L}$-interpretation of the extended set of prefixes $\mathcal{P}_\sigma(s')$ into $\langle G, R, D, \bar{D}, \Vdash\!\!- \rangle$, *i.e.*, that for $q, r \in \mathcal{P}_\sigma(s')$

$$q \; R_0 \; r \quad \text{implies} \quad \iota'(q) \; R \; \iota'(r) \, .$$

By hypothesis, and the fact that ι agrees with ι' on $\mathcal{P}_\sigma(s)$, the above holds when both q and r are elements of $\mathcal{P}_\sigma(s)$. Moreover, Lemma 6.29 tells us that there are no $r \in \mathcal{P}_\sigma(s)$, with $p\pi_0 \; R_0 \; r$ (S5: $\pi_0 \; R_0 \; r$). We have, therefore, only to deal with the case of $q \in \mathcal{P}_\sigma(s)$ for which $q \; R_0 \; p\pi_0$ (S5: $q \; R_0 \; \pi_0$).

For S5:

$$\iota'(q) \; R \; \iota'(\pi_0)$$

since R ensures that every point of G is accessible from every other point.

For the other logics, suppose that $q \; R_0 \; p\pi_0$ for some $q \in \mathcal{P}_\sigma(s')$. There are a number of possibilities for q depending on the accessibility relation on prefixes for that logic (Table 6–2). We examine each condition in turn.

1. $q = p$. (General: all logics except S5.) Notice that:

   $$\iota'(q) = \iota(q) = \iota(p) = \iota'(p)$$

   since $q = p$ and, $p \in \mathcal{P}_\sigma(s)$. Also $\iota'(p) \; R \; \iota'(p\pi_0)$ since $\iota'(p) \; R \; w$ and $w = p\pi_0$ (by construction). Hence:

   $$\iota'(q) \; R \; \iota'(p\pi_0).$$

2. $q = p\pi_0$. (Reflexive: T, S4.) Impossible, since by Lemma 6.29, $p\pi_0 \notin \mathcal{P}_\sigma(s)$.

3. $q \prec p$. (Transitive: K4, D4, S4.) Then:

   $$\iota'(q) \; R \; \iota'(p)$$

since $p, q \in \mathcal{P}_\sigma(s)$ and ι and ι' agree on $\mathcal{P}_\sigma(s)$. But:

$$\iota'(p)\ R\ \iota'(p\pi_0),$$

since $\iota'(p)\ R\ w$ and $w = p\pi_0$ by construction. Consequently:

$$\iota'(q)\ R\ \iota'(p\pi_0)$$

follows from the transitivity of R.

■

PROPOSITION 6.31 *Let s be a $\lhd$-compatible path through X^μ, $\mathcal{L}$-satisfiable under σ, and properly reducible on ν. The path obtained by reduction on ν is $\mathcal{L}$-satisfiable under σ.*

PROOF. Let k^κ be the ν-type position in question and l the child of k in the (unindexed) formula tree. By definition, the secondary type of l is ν_0. k^κ has $\mu(l)$ children in the indexed formula tree X^μ. For s to be properly reducible on k^κ it means that there is at least one j, $1 \leq j \leq \mu(l)$ such that $l^{\kappa j} \in \mathcal{D}(s)$. Furthermore, this position must be a $\lhd$-least element of $\mathcal{D}(s)$. Let ν_0 denote the position $l^{\kappa j}$. We continue as in the π case above.

The path obtained by reduction on ν is s, ν_0. Call this path s'. Let p be the image of the prefix of ν under σ, and q the image of ν_0 itself under σ; *i.e.*, $p = \sigma(\mathrm{pre}(\nu))$ and $q = \sigma(\nu_0)$. We have:

- $S(s') = S(s) \cup \{\nu_0\}$.
- $\mathcal{P}_\sigma(s') = \mathcal{P}_\sigma(s) \cup \{\, r \mid r \preceq pq \,\}$. (S5: $\mathcal{P}_\sigma(s') = \mathcal{P}_\sigma(s) \cup \{q\}$.)
- $C_\sigma(s') = C_\sigma(s)$.

Let Y be the image of $\mathrm{sform}(\nu)$ under σ. Note that Y is a signed formula of ν type. We shall use Y_0 to denote its immediate subformula; *i.e.*,

$$Y_0 = \sigma(\mathrm{sform}(\nu_0))\,.$$

By hypothesis we have the existence of an $\mathcal{L}$-model, $\langle G, R, D, \bar{D}, \Vdash\!\!\!- \rangle$, and an $\mathcal{L}$-interpretation, ι, for which:

- All constants of $\iota(Y)$ are in $\bar{D}(\iota(p))$, and: $\iota(p) \Vdash \iota(Y)$.
- For all $v \in C_\sigma(s)$ with $\sigma(\mathrm{pre}(v)) = r$, if $r \in P_\sigma(s)$, $\iota(v) \in \bar{D}(\iota(r))$.

Hence, by the model conditions (Corollary 4.1) for all points $w \in G$, with $\iota(p)\ R\ w$:

$$w \Vdash \iota(Y_0)\,.$$

If we can extend ι to an $\mathcal{L}$-interpretation ι' for $P_\sigma(s')$ such that

- $\iota'(p)\ R\ \iota'(pq)$, and
- if $v \in C_\sigma(s')$ and $\sigma(\mathrm{pre}(v)) = r$, then $r \in P_\sigma(s')$;

we are done, since, by the above model condition:

$$\iota'(pq) \Vdash \iota'(Y_0)\,,$$

i.e.,

$$\iota'(pq) \Vdash \sigma(\mathrm{sform}(\nu_0))\,.$$

We deal with the logics other than S5 first. For these logics, the prefix of ν_0 (under σ) is the sequence pq. We identify two cases:

1. $q = \emptyset$. (Reflexive logics: T and S4.) In this case p is the prefix of both ν_0 and ν under σ; *i.e.*, $P_\sigma(s') = P_\sigma(s)$. Define $\iota' = \iota$. Since ι' is defined on p, and R is reflexive: $\iota'(p)\ R\ \iota'(p)$. That is:

   $$\iota'(p)\ R\ \iota'(pq)\,.$$

   The Parameter Condition holds for s' under ι' by Lemma 6.20.

2. $q = u_1 u_2 \cdots u_n$, $1 \leq n$. We construct a series of $\mathcal{L}$-interpretations $\iota_0, \iota_1, \ldots, \iota_n$ for the sets: $P^i_\sigma(s)$ and $C^i_\sigma(s)$, $0 \leq i \leq n$, defined inductively as follows:

   $$\begin{aligned} P^0_\sigma(s) &= P_\sigma(s) \\ P^{i+1}_\sigma(s) &= P^i_\sigma(s) \cup \{\, pu_1 \cdots u_{i+1} \,\}\,, \quad 0 \leq i \leq n-1. \end{aligned}$$

   $$C^i_\sigma(s) = C_\sigma(s)\,, \quad 0 \leq i \leq n.$$

The construction of ι_n is inductive, and ensures that:

$$\iota_n(p)\ R\ \iota_n(pq)\,.$$

- Base case: we take $\iota_0 = \iota$. ι_0 is an $\mathcal{L}$-interpretation for $P^0_\sigma(s)$ by hypothesis.
- Inductive case: given ι_i, $0 \leq i < n-1$, we construct ι_{i+1} as follows. There are two cases depending on whether $pu_1 \cdots u_{i+1} \in P^i_\sigma(s)$ or not.

  (a) $pu_1 \cdots u_{i+1} \in P^i_\sigma(s)$. Then ι_i is already defined on $pu_1 \cdots u_{i+1}$, and, by the induction hypothesis, is an $\mathcal{L}$-interpretation for $P^i_\sigma(s)$. Define $\iota_{i+1} = \iota_i$. Since

  $$pu_1 \cdots u_i\ R_0\ pu_1 \cdots u_{i+1}$$

  we have:

  $$\iota_i(pu_1 \cdots u_i)\ R\ \iota_i(pu_1 \cdots u_{i+1})\,,$$

  and hence:

  $$\iota_{i+1}(pu_1 \cdots u_i)\ R\ \iota_{i+1}(pu_1 \cdots u_{i+1})\,.$$

  The Parameter Condition holds since we have: $P^{i+1}_\sigma(s) = P^i_\sigma(s)$, $C^{i+1}_\sigma(s) = C^i_\sigma(s)$ and finally $\iota_{i+1} = \iota_i$.

  (b) $pu_1 \cdots u_{i+1} \notin P^i_\sigma(s)$. This situation cannot arise in the case of the non-idealisable K-logics: K and K4, since by definition, properly reducing a ν_0-type position cannot introduce any new prefixes. For the others, since they are idealisable, there is a $w \in G$ with $\iota_i(pu_1 \cdots u_i)\ R\ w$. Extend the interpretation to $P^{i+1}_\sigma(s)$ by defining

  $$\iota_{i+1}(q') = \begin{cases} w, & q' = pu_1 \cdots u_{i+1} \\ \iota_i(q')\,, & q' \in P^i_\sigma(s)\,. \end{cases}$$

  We have to show that the extended mapping is indeed an $\mathcal{L}$-interpretation. There can be no $p' \in P^i_\sigma(s)$ with $pu_1 \cdots u_{i+1}\ R_0\ p'$. ($pu_1 \cdots u_{i+1}$ would be an initial sequence of such a p', and

all such initial sequences are in $P^i_\sigma(s)$. Hence $pu_1 \cdots u_{i+1} \in P^i_\sigma(s)$ contradicting our assumption that $pu_1 \cdots u_{i+1} \notin P^i_\sigma(s)$.) Suppose $p'\ R_0\ pu_1 \cdots u_{i+1}$. Then p' must be an initial sequence of $pu_1 \cdots u_{i+1}$. There are three possibilities depending on the form of the accessibility relation on prefixes for the logic (Table 6–2).

i. $p' = pu_1 \cdots u_i$. (General: D, D4, T, S4.) Since

$$\iota_i(pu_1 \cdots u_i)\ R\ w,$$

ι_i agrees with ι_{i+1} on $P^i_\sigma(s)$ and $w = \iota_{i+1}(pu_1 \cdots u_{i+1})$, we have:

$$\iota_{i+1}(pu_1 \cdots u_i)\ R\ \iota_{i+1}(pu_1 \cdots u_{i+1}).$$

ii. $p' = pu_1 \cdots u_{i+1}$. (Reflexive: T, S4.) Notice that

$$\iota_{i+1}(pu_1 \cdots u_{i+1})\ R\ \iota_{i+1}(pu_1 \cdots u_{i+1})$$

since R is reflexive.

iii. $p' \prec pu_1 \cdots u_i$. (Transitive: D4, S4.) In this case we have $p'\ R_0\ pu_1 \cdots u_i$, and hence $\iota_{i+1}(p')\ R\ \iota_{i+1}(pu_1 \cdots u_i)$ since both p' and $pu_1 \cdots u_i$ are members of $P^i_\sigma(s)$, and ι_i agrees with ι_{i+1} on $P^i_\sigma(s)$. By construction,

$$\iota_{i+1}(pu_1 \cdots u_i)\ R\ \iota_{i+1}(pu_1 \cdots u_{i+1}).$$

Hence, by the transitivity of R we have

$$\iota_{i+1}(p')\ R\ \iota_{i+1}(pu_1 \cdots u_{i+1})$$

as required.

The Parameter Condition holds in the varying and cumulative domain cases by hypothesis, and since $pu_1 \cdots u_{i+1} \notin P^i_\sigma(s)$ (Lemma 6.21).

This completes the inductive construction of $\iota_0, \iota_1, \ldots, \iota_n$, $1 \leq n$. We have that

$$\iota_n(pu_1 \cdots u_i)\ R\ \iota_n(pu_1 \cdots u_{i+1}), \qquad 0 \leq i < n-1.$$

Recall that $q = u_1 \cdots u_n$. For the non-transitive logics: K, D, T, $n \leq 1$, and hence

$$\iota_n(p)\ R\ \iota_n(pq)\,.$$

For the transitive logics: K4, D4, S4, the transitivity of R gives us:

$$\iota_n(p)\ R\ \iota_n(pq)\,.$$

The Parameter Condition also holds for s' under ι' by construction. Hence s' is $\mathcal{L}$-satisfiable under σ.

This finishes the treatment of the logics other than S5. For S5 the argument is simple. If $p = \sigma(\mathrm{pre}(\nu))$ and $q = \sigma(\mathrm{pre}(\nu_0))$ then $p, q \in T_M(\mu)$ (*i.e.*, p and q are positions). There are two possibilities depending on whether q is already in $P_\sigma(s)$ or not.

1. $q \in P_\sigma(s)$. Then ι is defined on q. Since every point is accessible from every other point, we have

   $$\iota(p)\ R\ \iota(q)\,.$$

   The Parameter Condition holds by hypothesis.

2. $q \notin P_\sigma(s)$. Choose $w \in G$ with $\iota(p)\ R\ w$, and extend ι to $P_\sigma(s')$ by defining:

   $$\iota'(r) = \begin{cases} w, & r = q \\ \iota(r), & \text{otherwise.} \end{cases}$$

   Again, since every point is accessible from every other point,

   $$\iota(p)\ R\ \iota(q)\,,$$

   and the extended mapping is an S5-interpretation for $P_\sigma(s')$. The Parameter Condition holds for varying domains by hypothesis, and by Lemma 6.21. It holds for constant domains of course (Lemma 6.19).

By hypothesis,

$$\iota(p) \Vdash \iota(\sigma(\mathrm{sform}(\nu)))\,.$$

Since $\iota(p)\ R\ \iota(q)$, we have

$$\iota(q) \Vdash \iota(\sigma(\mathrm{sform}(\nu_0)))$$

by the model conditions (Corollary 4.1). Hence s' is S5-satisfiable under σ.

■

6.3.4 Summary.

We are now in a position to prove the correctness of the matrix characterisations.

THEOREM 6.32 (CORRECTNESS) *A modal formula A is $\mathcal{L}$-valid if there is a modal multiplicity μ, an $\mathcal{L}$-admissible substitution σ, and a set of σ-complementary connections that spans the indexed formula $\langle A, 0\rangle^{\mu}$.*

PROOF. Suppose not. That is, suppose the multiplicity, substitution and spanning connection set exist, but A is not $\mathcal{L}$-valid. Let C_0 denote the constants of A. Then there is an $\mathcal{L}$-model, $\left\langle G, R, D, \bar{D}, \Vdash \right\rangle$, and an interpretation, ι, in the model, and some $w \in G$ with $\iota(a) \in \bar{D}(w)$, for every $a \in C_0$, such that:

$$w \nVdash \iota(A)$$

(equivalently: $w \Vdash \iota(\langle A, 0\rangle)$).

For the root path $s_0 = \{k_0\}$ (k_0 is the root position of the formula tree for $\langle A, 0\rangle$) we have:

- $S(s_0) = \{\, k_0 \,\}$.
- $P_\sigma(s_0) = \{\, k_0 \,\}$.
- $C_\sigma(s_0) = C_0$.

Extend the mapping ι by putting: $\iota(k_0) = w$. ι is then an $\mathcal{L}$-interpretation for $P_\sigma(s_0)$ and $C_\sigma(s_0)$. Furthermore, since

$$\sigma(\mathrm{pre}(k_0)) = k_0 \quad \text{and} \quad \sigma(\mathrm{sform}(k_0)) = \langle A, 0\rangle$$

we have:

$$\iota(k_0) \parallel\!\!- \sigma(\mathrm{sform}(k_0))\,,$$

by construction. The root path is thus $\mathcal{L}$-satisfiable under σ.

For the idealisable logics D, D4, T, S4, S5, we can reduce the root path to the set of atomic paths through $\langle A, 0\rangle^{\mu}$ by a series of proper reductions (Proposition 6.14). For the non-idealisable logics K and K4, we can reduce the root path to the set of K-atomic paths through $\langle A, 0\rangle^{\mu}$ by a series of proper reductions (Proposition 6.14).

Since the root path is $\mathcal{L}$-satisfiable under σ, at least one of the atomic paths (K-atomic paths) through $\langle A, 0\rangle^{\mu}$ is $\mathcal{L}$-satisfiable under σ, by Propositions 6.23, 6.24, 6.26, 6.27, 6.30 and 6.31.

But every atomic path contains a σ-complementary connection, which contradicts Proposition 6.22; directly in the case of the idealisable logics, and with the use of Proposition 6.16 in the case of the non-idealisable logics. Hence, A is valid. ■

6.4 Completeness.

In this section we prove that the matrix characterisations of validity presented above are complete for the modal logics under consideration. Once again our methods follow the pattern of standard *systematic* proofs of completeness for analytic tableau and sequent-based proof systems. The systematic nature of the proof defines semi-decision procedures for the (first-order) logics. In the next chapter we outline more efficient methods of proof search based on the matrix characterisations.

For the rest of this section we assume that A is a modal sentence and X the signed formula $\langle A, 0\rangle$. Under these assumptions, the statement of completeness for the matrix characterisations is:

if A is $\mathcal{L}$-valid, there is a multiplicity, μ, for X, an $\mathcal{L}$-admissible substitution, σ, for X^μ and a set of σ-complementary connections that span X^μ.

6.4.1 Overview.

The idea behind the proofs is simple. We define a procedure guaranteed to construct a multiplicity, μ, and an $\mathcal{L}$-admissible substitution, σ, such that either:

(a) there exists an atomic path through X^μ which is $\mathcal{L}$-satisfiable under σ; or

(b) every atomic path through X^μ contains a σ-complementary connection.

For any multiplicity and $\mathcal{L}$-admissible substitution, the root position, k_0, of the indexed formula is contained in the associated set of any path. Moreover, the polarity of k_0 is 0. Hence the first conclusion entails that the label of k_0, *i.e.*, A, cannot be $\mathcal{L}$-valid. The $\mathcal{L}$-validity of A then forces the second conclusion, and hence completeness.

The two central issues are therefore:

- the nature of such a multiplicity and substitution; and
- the definition of the procedure itself.

We deal with each issue, in that order, in the next two sections.

6.4.2 $\mathcal{L}$-Hintikka sets and $\mathcal{L}$-Complete paths.

First we define a basic $\mathcal{L}$-satisfiable structure called an $\mathcal{L}$-Hintikka set, comprising signed formulae and prefixes.

Let $\langle G_0, R_0, D_0, \bar{D}_0 \rangle$ be a first-order $\mathcal{L}$-frame. Furthermore, suppose the set of constants D_0 is partitioned into sets: $D_0(p)$ for $p \in G_0$, such that D_0 is the union of all the $D_0(p)$. That is to say,

$$D_0 = \bigcup_{p \in G_0} D_0(p).$$

Let $P_0 \subseteq G_0$ and $C_0 \subseteq D_0$. A *prefixed* signed formula over P_0 and C_0 is a pair, $p : Y$, where $p \in P_0$ and Y is a signed formula over C_0. We say that a set, S, of prefixed signed formulae over P_0 and C_0, is $\mathcal{L}$-satisfiable just in case there is an $\mathcal{L}$-model $\langle G, R, D, \tilde{D}, \Vdash \rangle$, and an $\mathcal{L}$-interpretation, ι, of P_0 and C_0 in the model, such that: for each $p : Y \in S$, all the constants of $\iota(Y)$ are in $\tilde{D}(\iota(p))$ and

$$\iota(p) \Vdash \iota(Y) .$$

For a set, S, of prefixed signed formula over P_0 and C_0, define $C_0(p)$, for $p \in P_0$, as follows:

$$C_0(p) \stackrel{\text{df}}{=} C_0 \cap D_0(p).$$

That is, $C_0(p)$ is the set of constants of (the prefixed formulae of) S associated with the prefix p. We say that S is an $\mathcal{L}$-Hintikka set (of prefixed signed formulae) just in case:

0. There is no atomic formula B for which, for some $p \in P_0$, both

$$p : \langle B, 1 \rangle \in S \quad \text{and} \quad p : \langle B, 0 \rangle \in S.$$

1. $p : \alpha \in S$ implies $p : \alpha_1 \in S$ and $p : \alpha_2 \in S$.

2. $p : \beta \in S$ implies $p : \beta_1 \in S$ or $p : \beta_2 \in S$.

3. $p : \gamma \in S$ implies $p : \gamma_0(a) \in S$ for all a such that:

   Constant domains: $a \in C_0$.

   Varying domains: $a \in C_0(p)$.

   Cumulative domains: $a \in C_0(q)$, for some $q \in P_0$ with either (a) $q = p$, or (b) $q \; R_0 \; p$.

4. $p : \delta \in S$ implies $p : \delta_0(a) \in S$ for some $a \in C_0(p)$.

5. $p : \nu \in S$ implies $q : \nu_0 \in S$, for all $q \in P_0$ such that $p \; R_0 \; q$, and, for the D-logics only: there is such a q.

6. $p : \pi \in S$ implies $q : \pi_0 \in S$, for some $q \in P_0$ such that $p\ R_0\ q$.

The import of the notion of an $\mathcal{L}$-Hintikka set is summarised by the following theorem, a version of which is proved in [Fit83]:

THEOREM 6.33 (FITTING [FIT83]) *Any $\mathcal{L}$-Hintikka set over P_0 and C_0 is $\mathcal{L}$-satisfiable in a model $\langle P_0, R_0, C_0, \bar{C}_0, \Vdash\!\!- \rangle$, under the identity $\mathcal{L}$-interpretation, where $C_0(p) \subseteq \bar{C}_0(p)$.*

REMARK. What we call $\mathcal{L}$-Hintikka sets, Fitting calls *$\mathcal{L}$-downward saturated sets*. We use the former for brevity. (END OF REMARK.)

Next we define a property of a multiplicity and $\mathcal{L}$-admissible substitution that extends this $\mathcal{L}$-satisfiable structure to paths and positions.

Let μ be a multiplicity for X and σ an $\mathcal{L}$-admissible substitution for X^μ. Recall that, for any path s, $C_\sigma(s)$ denotes the set of constants of the labels of the set of positions $S(s)$, and $P_\sigma(s)$ denotes the set of prefixes of these positions. We define the set of constants *associated* with the prefix $p \in P_\sigma(s)$, denoted $C_\sigma(s,p)$, as follows:

$$C_\sigma(s,p) \stackrel{\text{df}}{=} \{\, v \in C_\sigma(s) \mid \sigma(\text{pre}(v)) = p \,\}.$$

An atomic path, s, through X^μ is said to be *$\mathcal{L}$-complete under σ* just in case:

1. For all $\gamma^\kappa \in s$ with $p = \sigma(\text{pre}(\gamma^\kappa))$ and for every v such that:

   Constant domains: $v \in C_\sigma(s)$;

   Varying domains: $v \in C_\sigma(s,p)$;

   Cumulative domains: $v \in C_\sigma(s,q)$, for some $q \in P_\sigma(s)$ with either (a) $q = p$, or (b) $q\ R_0\ p$;

   there exists a j, $0 \leq j \leq \mu(\gamma_0)$, such that:

$$\sigma(\gamma_0^{\kappa j}) = v.$$

2. For all $\nu^\kappa \in s$ with $p = \sigma(\mathrm{pre}(\nu^\kappa))$ and for every $q \in \mathcal{P}_\sigma(s)$ with $p\ R_0\ q$, there exists a j, $0 \leq j \leq \mu(\nu_0)$, such that:

$$\sigma(\mathrm{pre}(\nu_0^{\kappa j})) = q.$$

The conditions here for elements of Γ and $\mathcal{V}$ should be compared with the conditions required for prefixed signed formulae of γ-type and $\mathcal{V}$ -type respectively in the definition of an $\mathcal{L}$-Hintikka set.

Let μ be a multiplicity for X and σ an $\mathcal{L}$-admissible substitution for X^μ. Define the set of prefixed signed formulae, Pforms(s), as follows:

$$\mathrm{Pforms}(s) \stackrel{\mathrm{df}}{=} \{\, p : Y \mid p = \sigma(\mathrm{pre}(u)), Y = \sigma(\mathrm{sform}(u)), u \in S(s) \,\}.$$

Notice that Pforms(s) is a set of prefixed signed formulae over the set of prefixes: $\mathcal{P}_\sigma(s)$, and the set of constants: $\mathcal{C}_\sigma(s)$. Notice also that the prefixed signed formula:

$$k_0 : \langle A, 0 \rangle,$$

is an element of Pforms(s), for every path s. We now have:

LEMMA 6.34 *Let s be a non-complementary atomic path through X^μ. If s is $\mathcal{L}$-complete under σ, Pforms(s) is an $\mathcal{L}$-Hintikka set.*

PROOF. The correspondence between the definitions of $\mathcal{L}$-complete and $\mathcal{L}$-Hintikka set are obvious: $\mathcal{P}_\sigma(s) = P_0$, $\mathcal{C}_\sigma(s) = C_0$ and $\mathcal{C}_\sigma(s, p) = C_0(p)$.

First notice that the atomic clause (0) is satisfied by virtue of s being non-complementary.

Secondly notice that the propositional clauses: (1) and (2), dealing with atomic, α and β type formulae are satisfied by virtue of s being an atomic path.

Next notice that the δ and π clauses: (4) and (6), are satisfied, by the definition of labels in the first case, and the definition of prefixes in the second, together with the fact that s is an atomic path. (Note that the

condition that $a \in C_0(p)$ is satisfied under the correspondence of $C_\sigma(s,p)$ with $C_0(p)$ and the definition of the former.

Finally, by the definition of paths, all γ and ν-type positions of $S(s)$ are elements of s itself. (They are never "deleted," so to speak, by a path reduction.) Under the correspondence outlined above, conditions (3) and (5) for γ and ν-type prefixed signed formulae respectively are satisfied by the conditions stipulated for γ and ν-type positions in the definition of the $\mathcal{L}$-completen ss of a path. ■

Recall the definition of when a path s through X^μ is $\mathcal{L}$-satisfiable under σ from the previous section: we need an $\mathcal{L}$-model, $\langle G, R, D, \bar{D}, \|\!\!- \rangle$, and an $\mathcal{L}$-interpretation, ι, of $P_\sigma(s)$ and $C_\sigma(s)$ in the model such that:

1. For each $u \in S(s)$, with $p = \sigma(\mathrm{pre}(u))$ and $Y = \sigma(\mathrm{sform}(u))$, all the constants of $\iota(Y)$ are in $\bar{D}(\iota(p))$ and:

$$\iota(p) \|\!\!- \iota(Y)\,.$$

2. For each $v \in C_\sigma(s)$ with $q = \sigma(\mathrm{pre}(v))$:

$$q \in P_\sigma(s) \quad \text{implies} \quad \iota(v) \in \bar{D}(\iota(q)).$$

We now have the central proposition:

PROPOSITION 6.35 *Let s be a non-complementary atomic path through X^μ. If s is $\mathcal{L}$-complete under σ, it is $\mathcal{L}$-satisfiable under σ.*

PROOF. Immediate from Theorem 6.33 and Lemma 6.34. Note that the Paramater Condition is satisfied by the definition of $C_\sigma(s,p)$, its correspondence with $C_0(p)$ in Lemma 6.34, and finally the particular model asserted to exist by Theorem 6.33. ■

From this we deduce:

COROLLARY 6.36 *For a modal formula A, if there exists an multiplicity, μ, an $\mathcal{L}$-admissible substitution, σ, and an atomic path, s, through $\langle A,0\rangle^{\mu}$ which is $\mathcal{L}$-complete under σ, then A is not valid.*

PROOF. Under the hypotheses, the previous proposition tells us that the atomic path is $\mathcal{L}$-satisfiable under σ. This means there is an $\mathcal{L}$-model, $\langle G, R, D, \bar{D}, \Vdash\!\!-\rangle$, and an $\mathcal{L}$-interpetation, ι, with the usual properties with respect to the positions of $S(s)$. In particular, since $k_0 \in S(s)$ we have (by the first condition of $\mathcal{L}$-satisfiability under σ): all constants of A are in $\bar{D}(\iota(k_0))$ and

$$\iota(k_0) \Vdash\!\!- \iota(\langle A,0\rangle)\,.$$

Since:

- k_0 is the prefix of k_0.
- $k_0 \in \Pi_0(\mu)$, *i.e.*, it is constant under σ.
- $\mathrm{sform}(k_0) = \langle A,0\rangle$.
- A contains no "parameter" positions so $\mathrm{sform}(k_0)$ is constant under σ.

Consequently, A is not valid. ■

This achieves our first goal, namely to capture the nature of a multiplicity μ and $\mathcal{L}$-admissible substitution σ that ensures that a non-complementary atomic path is $\mathcal{L}$-satisfiable under σ. In the next section we present a procedure that constructs such a pair $\langle \mu, \sigma\rangle$ for any non-valid modal sentence.

6.4.3 The systematic procedure.

In this section we define a procedure which, for a modal formula A, constructs a multiplicity, μ, and an $\mathcal{L}$-admissible substitution, σ, such that either:

(a) there exists an atomic path through $\langle A,0\rangle^{\mu}$ which is $\mathcal{L}$-complete under σ;
or

(b) every atomic path through $\langle A, 0\rangle^{\mu}$ contains a σ-complementary connection.

By Corollary 6.36, the first option would entail that A is not valid. Therefore, if A is valid, the procedure must terminate with the second option. Hence completeness.

Recall that the conditions on modal substitutions were:

A. $u \varepsilon \sigma(v)$ implies $\sigma(u) = u$.

B. $pu \in P_{\sigma}(\mu)$ implies $pu = \sigma(\mathrm{pre}(u))$.

The conditions for $\mathcal{L}$-admissibility were:

1. σ respects $\mathcal{L}$-accessibility relations R_0, *i.e.*, for all $p, q \in T_M^*$,

$$p\, R_0\, q \quad \text{implies} \quad \sigma(p)\, R_0\, \sigma(q)$$

2. (K-logics only) $u \sim u'$ implies $v \sqsubset u$ for some α-related position v.

3. $\lhd = (\ll \cup \sqsubset_M \cup \sqsubset_Q)^+$ is irreflexive.

4. The following condition holds for σ_M and σ_Q depending on the domain condition for the logic $\mathcal{L}$:

   Constant domains: No condition.

   Varying domains: If $\sigma_Q(u') = v'$, then $\sigma_M^{\#}(\mathrm{pre}(v')) = \sigma_M^{\#}(\mathrm{pre}(u'))$.

   Cumulative domains: If $\sigma_Q(u') = v'$, then either:

   (a) $\sigma_M^{\#}(\mathrm{pre}(v')) = \sigma_M^{\#}(\mathrm{pre}(u'))$; or

   (b) $\sigma_M^{\#}(\mathrm{pre}(v'))\ R_0\ \sigma_M^{\#}(\mathrm{pre}(u'))$.

In what follows, we refer to the four conditions for $\mathcal{L}$-admissibility above as the "First," "Second," "Third" and "Fourth" conditions respectively.

Let $\mathcal{A}(\mu)$ denote the set of atomic paths through X^{μ}. We construct inductively a sequence $\langle \mu_0, \sigma_0\rangle, \langle \mu_1, \sigma_1\rangle, \ldots, \langle \mu_n, \sigma_n\rangle, \ldots$ of pairs of multiplicities and $\mathcal{L}$-admissible mappings as follows:

Base. Define μ_0 to be the constant zero function; *i.e.*, for all $u \in \mathcal{V}_0$, $\mu_0(u) = 0$, and σ_0 to be the identity function on $T_Q(\mu_0)$ and $T_M(\mu_0)$. We claim first that σ_0 is an $\mathcal{L}$-admissible substitution.

PROOF. Condition A clearly holds for the identity function. Condition B holds by the definition of prefixes. Consequently σ_0 is a modal substitution. The identity function clearly satisfies the First condition. Since $\Gamma_0(\mu_0) = \mathcal{V}_0(\mu_0) = \emptyset$, $\sim_Q$, $\sqsubset_Q$, $\sim_M$ and $\sqsubset_M$, are the empty relations. Consequently the Second, Third and Fourth conditions are also satisfied. Hence σ_0 is $\mathcal{L}$-admissible. ■

Induction. Having completed stage n, if there is a set of σ-complementary connections that spans $\mathcal{A}(\mu_n)$, stop. Likewise, if any non-complementary path in $\mathcal{A}(\mu_n)$ is $\mathcal{L}$-complete under σ_n, stop. (We note that the test for $\mathcal{L}$-completeness under a substitution is effective.)

Otherwise construct $\langle \mu_{n+1}, \sigma_{n+1} \rangle$ as follows: from the non-complementary atomic paths in $\mathcal{A}(\mu_n)$ select a path s with smallest $S(s)$. From s select the first (or leftmost) non-atomic position, say l^κ. (Here we assume that the non-atomic positions of an atomic path are ordered in some uniform fashion.) Note that l^κ is either of γ or ν type, since these are the only non-atomic positions of an atomic path. Let p denote the image of the prefix of l^κ under σ_n, and l_0 the child of l in the unindexed formula tree. We deal with each case in turn.

1. $l^\kappa \in \Gamma(\mu_n)$. Then $l_0 \in T_Q$. Let Q be the largest set defined according to the domain condition of $\mathcal{L}$ as follows:

   Constant domains: $Q \subseteq C_\sigma(s)$;

   Varying domains: $Q \subseteq C_\sigma(s, p)$;

   Cumulative domains: $Q \subseteq \bigcup_{q \in P_\sigma(s)} C_\sigma(s, q)$ such that either: (a) $q = p$, or (b) $q \; R_0 \; p$;

   such that for all $u \in Q$, there is no j, $1 \leq j \leq \mu_n(l_0)$, with:

$$\sigma(l_0^{\kappa\, j}) = u.$$

There are two cases depending on whether Q is empty or not.

(a) $Q = \emptyset$. This case occurs if l^κ is "complete" already. Define $\mu_{n+1} = \mu_n$, and $\sigma_{n+1} = \sigma_n$. σ_{n+1} is therefore $\mathcal{L}$-admissible by hypothesis.

(b) $Q = \{c_1, c_2, \ldots, c_m\}$, for $1 \leq m$. Define μ_{n+1} as follows:

$$\mu_{n+1}(v) = \begin{cases} \mu_n(v) + m, & v = l_0 \\ \mu_n(v), & v \neq l_0. \end{cases}$$

Define σ_{n+1} as follows:

$$\sigma_{n+1}(v) = \begin{cases} c_i, & v = l_0^{\kappa j},\ \mu_n(l_0) \leq j \leq \mu_{n+1}(l_0),\ i = j - \mu_n(l_0); \\ \sigma_n(v), & \text{otherwise.} \end{cases}$$

We claim that σ_{n+1} is $\mathcal{L}$-admissible.

PROOF. the modal component of the substitution is unchanged. Hence the First, Second and Third conditions are satisfied. Notice that the $l_0^{\kappa j}$ are all new γ_0-type positions. Consequently, the reduction relation induced by the extended substitution remains irreflexive. The construction was arranged such that the Fourth condition holds. ■

This completes the construction for a γ type position.

2. $l^\kappa \in \mathcal{V}(\mu_n)$. Let Q denote the set of prefixes $q \in P_\sigma(s)$ with $p\ R_0\ q$ and no j, $1 \leq j \leq \mu_n(l_0)$, such that:

K, K4, D, D4, T, S4: $pr = q$, where $r = \sigma_n(l_0^{\kappa j})$.

S5: $q = \sigma_n(l_0^{\kappa j})$.

Again there are two cases depending on whether or not Q is empty:

(a) $Q = \emptyset$. For the logics other than the D-logics define

$$\mu_{n+1} = \mu_n \qquad \text{and} \qquad \sigma_{n+1} = \sigma_n.$$

σ_{n+1} is $\mathcal{L}$-admissible by hypothesis.

For the D-logics define μ_{n+1} as follows:

$$\mu_{n+1}(v) = \begin{cases} \mu_n(v) + 1, & v = l_0 \\ \mu_n(v), & v \neq l_0. \end{cases}$$

Let $i = \mu_n(l_0) + 1$. For the D-logics define σ_{n+1} as follows:

$$\sigma_{n+1}(v) = \begin{cases} l_0^{\kappa i}, & v = l_0^{\kappa i} \\ \sigma_n(v), & v \neq l_0^{\kappa i}. \end{cases}$$

We claim that the extended mapping σ_{n+1} is an $\mathcal{L}$-admissible substitution.

PROOF. First notice that $l_0^{\kappa i}$ is completely new. Hence its prefix: $pl_0^{\kappa i}$, is the only new prefix and Conditions A and B are satisfied by construction. It is easy to check that σ_{n+1} is $\mathcal{L}$-admissible by virtue of the fact that $l_0^{\kappa i}$ is new. (Note that accessibility on prefixes for all of the logics satisfies the general condition.) ■

(b) $Q \neq \emptyset$. Suppose Q has m elements $q_1, q_2, \ldots, q_m$, $(1 \leq m)$. Define μ_{n+1} as follows:

$$\mu_{n+1}(v) = \begin{cases} \mu_n(v) + m, & v = u_0 \\ \mu_n(v), & v \neq u_0. \end{cases}$$

Suppose, without loss of generality that $r_1, \ldots, r_m$ are elements of $T_M(\mu_n)^*$ such that:

$$q_i = pr_i, \quad 1 \leq i \leq m.$$

Define σ_{n+1} as follows:

$$\sigma_{n+1}(v) = \begin{cases} r_i, & v = l_0^{\kappa j},\ \mu_n(l_0) \leq j \leq \mu_{n+1}(l_0),\ i = j - \mu_n(l_0) \\ \sigma_n(v), & \text{otherwise.} \end{cases}$$

We have to show that σ_{n+1} is an $\mathcal{L}$-admissible substitution. σ_{n+1} is a modal substitution since:

A. $l_0^{\kappa j} \notin |\mathcal{P}_\sigma(s)|$, $\mu_n(l_0) \leq j \leq \mu_{n+1}(l_0)$, and $v \in q_i$, $1 \leq i \leq m$, implies $v \in |\mathcal{P}_\sigma(s)|$. Since the condition held for σ_n, it holds for σ_{n+1}.

B. By construction and the fact that σ_n was a modal substitution.

σ_{n+1} is $\mathcal{L}$-admissible since:

1. σ_{n+1} respects $\mathcal{L}$-accessibility relations by construction.
2. (K-logics). $v \varepsilon r_i$, $1 \leq i \leq m$ implies $v \varepsilon |P_\sigma(s)|$. Hence all such positions are α-related to the $l_0^{\kappa j}$. The existence of at least one is guaranteed by the $\mathcal{L}$-admissibility of σ_n.
3. $\lhd$ remains irreflexive since the $l_0^{\kappa j}$, $\mu_n(l_0) \leq j \leq \mu_{n+1}(l_0)$ are completely new.
4. The first-order substitution has not altered so the fourth condition holds.

We now mark l^κ as the rightmost element of all the new atomic paths in $\mathcal{A}(\mu_{n+1})$ obtained by virtue of any increase in multiplicity. If no new paths result, l^κ is marked as the rightmost element of s in $\mathcal{A}(\mu_{n+1})$. This completes the construction.

The procedure given above is a matrix-based analogue of the usual systematic construction of tableau or sequent proofs. We can now prove completeness.

THEOREM 6.37 (COMPLETENESS) *If A is an $\mathcal{L}$-valid modal formula, there is a multiplicity μ, an $\mathcal{L}$-admissible substitution σ, and a set of σ-complementary connections that spans $\langle A, 0\rangle^\mu$.*

PROOF. Perform the systematic construction given above. There are the following possibilities:

1. The procedure terminates with a multiplicity μ, $\mathcal{L}$-admissible substitution σ, and either
   (a) a set of σ-complementary connections that span X^μ; or
   (b) a non-complementary atomic path through X^μ, $\mathcal{L}$-complete under σ.
2. The procedure fails to terminate and hence there is a path $s \in \mathcal{A}(\mu)$ for which $S(s)$ is infinite. (This conclusion requires the familiar use of Königs Lemma for infinite, finite branching trees.) By the systematic nature of the construction, s is $\mathcal{L}$-complete under σ. That is, we have

arranged things so that every atomic path is periodically considered (it becomes a smallest such path) and every position in it considered in turn (under our assumption that the elements of the paths have been ordered in some fashion). Each time a position is considered it is "completed."

By Proposition 6.35, both (1b) and (2) are impossible, hence the result. ■

6.5 Conclusions.

In this chapter we have presented matrix-based characterisations of validity for the modal logics K, K4, D, D4, T, S4 and S5. Our methods extend to the first-order systems including constant, varying and cumulative domain variants. We have proved the correctness and completeness of these characterisations.

The development was motivated by the proof-theoretic analysis of modal sequent calculi contained in the previous chapter, and the analysis of a matrix-based characterisation of validity for classical logic presented in the first part of the thesis. We identified a number of redundancies within the modal sequent calculi which we classified under the headings:

- notational redundancy,
- relevance, and
- order dependence.

To overcome similar problems for classical logic, Bibel [Bib82a] and Andrews [And81] used the notions of matrix, path and connection, developing a suitable notion of when two atomic formulae occurrences could be considered complementary. They reduced checking the validity of a sentence of first-order classical logic to a process of path checking and complementarity tests performed by a unification algorithm.

The material of this chapter represents a comprehensive extension of these ideas to modal logics. As in the classical case, valid formulae are characterised by their syntactic structure. Indeed, so uniform is the extension that search strategies based on the classical characterisation are immediately applicable in the modal case. We have sucessfully reduced checking a modal formula for validity in a modal logic to a process of path checking and complementarity tests performed by a specialised unification algorithm reflecting the properties of the accessibility relation for that logic.

In the next chapter we outline efficient proof systems based on the matrix characterisations, and discuss their use as decision procedures for the propositional fragments of the logics.

Chapter 7

Matrix-based proof search in modal logics.

7.1 Introduction.

In Chapter 5 we identified certain redundancies within the search space generated by the modal sequent calculi of Chapter 4. These redundancies were shown to render the sequent systems inefficient for automated proof search. In the previous chapter, inspired by our analysis of classical logic presented in Part I of this thesis, we developed matrix-based characterisations of validity for modal logics. In this chapter we investigate aspects of proof search based on the matrix characterisations, and demonstrate, amongst other things, that it is free of the aforementioned redundancies.

An important aspect of matrix-based proof search in modal logic is the proposed use of specialised unification algorithms to determine the complementarity of pairs of atomic formula occurrences. In §7.2 we discuss the properties of these unification problems and show that they are tractable variants of common unification problems under equational theories. Next, in §7.3, we present a series of examples designed to show that matrix-based proof search is free from the redundancies demonstrated to arise within sequent-based proof search. We also take the opportunity to illustrate some of the more subtle conditions placed on

$\mathcal{L}$-admissible substitutions that reflect the properties of various logics. In §7.4, we show how the matrix characterisations can be used to formulate efficient decision procedures for the propositional fragments of modal logics. For simplicity, we concentrate on S5. Finally, in §7.5, we deal with two miscellaneous but important issues. The first concerns the use of the matrix characterisations to decide instances of the standard consequence relation for modal logics. The second concerns the extension of the characterisations to languages involving function symbols.

7.2 Unification problems.

In §7.3, we show that proof procedures based on the matrix characterisations do not suffer from the redundancies inherent in sequent-based search methods. In Chapter 8 we demonstrate the advantages of matrix-based proof search over other methods proposed in the literature for automated proof search in modal logics. In these analyses we suppose that the basic component of matrix-based proof search is the selection of potentially complementary connections and the elimination of atomic paths spanned by these connections.

Recall that matrix characterisations of validity are couched in terms of the existence of $\mathcal{L}$-admissible substitutions that render a spanning set of connections simultaneously complementary. A search procedure must therefore ensure that after the addition of a connection to the current (non-spanning) set, there exists an $\mathcal{L}$-admissible substitution that makes all of the connections in the augmented set simultaneously complementary. We stress that the task is to ensure the existence of at least one such substitution, not to actually choose one.

The assumption then, is that the operation of deciding whether or not such an $\mathcal{L}$-admissible substitution exists after the addition of a new connection is a computationally tractable step. While the detailed development of individual proof procedures for modal logics based on the matrix characterisations is considered beyond the scope of this thesis, in this section we justify the above view of

matrix-based proof search in modal logics, by showing that the complementarity tests are indeed computationally tractable.

Our method is to show that the complementarity tests can be performed by refinements of standard unification algorithms. We do not persue an overly formal approach here, but merely seek to convince. In particular, we do not develop any unification algorithms specifically geared to the task at hand. The existence of more general algorithms in the literature is considered sufficient evidence that such specialised algorithms can be easily developed. In the last resort, one could simply apply these more general algorithms directly, then eliminate undesired substitutions.

The structure of this section is as follows: we begin with an overview to isolate the problematic components of the complementarity test (§7.2.1). We then prove some general properties of $\mathcal{L}$-admissible substitutions (§7.2.2), and proceed to treat each logic in turn thereafter.

7.2.1 Overview.

Let A be a modal formula and X the signed formula $\langle A, 0\rangle$. At each step of the search, we assume that the proof procedure maintains:

- a multiplicity for X;
- a set of connections U in X^{μ};
- a set of substitutions, denoted $\mathrm{Mgu}(U)$, for elements of $\Gamma_0(\mu)$ and $\nu_0(\mu)$, and
- an indication of the state of the basic atomic path checking search.

We require $\mathrm{Mgu}(U)$ to have the following properties:

1. Correctness. Every $\sigma \in \mathrm{Mgu}(U)$ is an $\mathcal{L}$-admissible substitution under which the elements of U are simultaneously σ-complementary.

2. Completeness. Every $\mathcal{L}$-admissible substitution under which the elements of U are simultaneously σ-complementary is an instance of some substitution of $\mathrm{Mgu}(U)$.

3. Minimality. No substitution in $\mathrm{Mgu}(U)$ is an instance of another distinct element of $\mathrm{Mgu}(U)$.

These conditions are precisely those defining the notion of a set of *most general unifiers* in the usual sense (see, for example, [Plo72,Sie84]).

Recall that, roughly speaking, an $\mathcal{L}$-admissible substitution σ renders a connection σ-complementary if it identifies the labels and prefixes of the atomic positions. Recall also that such substitutions comprise two components:

- a first-order substitution, σ_Q, that identifies the labels of the positions, and

- a modal substitution, σ_M, that identifies the prefixes of the positions.

(In the case of the varying and cumulative domain variants of the logics there are interactions between these components, but only in the sense that σ_Q may require prefixes other than those of the positions of the connection to be identified under σ_M.)

The calculation of a most general first-order substitution for the current set of connections poses no problem since we can employ Robinson's unification algorithm [Rob65] directly (or more efficient refinements of it). If any $\mathcal{L}$-admissible first-order substitutions exist at all for a given set of connections, there is a single most general one satisfying the conditions of correctness, completeness and minimality.

We can therefore structure $\mathrm{Mgu}(U)$ into two parts: $\mathrm{Mgu}_Q(U)$ and $\mathrm{Mgu}_M(U)$. $\mathrm{Mgu}_Q(U)$ comprises a single most general first-order substitution for the labels of the connections in U. $\mathrm{Mgu}_M(U)$ comprises a set of (independent) most general modal substitutions for U. Each element of $\mathrm{Mgu}_M(U)$ together with the sole element of $\mathrm{Mgu}_Q(U)$ forms an $\mathcal{L}$-admissible (combined) substitution.

The test of $\mathcal{L}$-admissibility for a combined substitution, comprises two major components:

- the calculation of the reduction ordering $\lhd$ and a test of its irreflexivity, and
- checking that the modal substitution respects the $\mathcal{L}$-accessibility relation on prefixes.

We remarked earlier that the irreflexivity test can be seen as a test of the acyclicity of a directed graph. Efficient algorithms for such operations are well known. The second test concerns only the modal substitution. We do not propose to generate modal substitutions and then test that they respect the $\mathcal{L}$-accessibility relation for the particular logic of interest. Rather, we suggest the use of specialised unification algorithms that only compute most general modal substitutions that satisfy this constraint. The constraint is effectively "built-in."

To summarise: we have argued that the only potentially problematic part of the complementarity test is the calculation of most general modal substitutions that respect the $\mathcal{L}$-accessibility relation for a given logic. In the following sections we argue that these calculations can be perfomed, respecting the constraints of correctness, completeness and minimality, by modifications of standard equational unification algorithms. We begin by noting some properties of $\mathcal{L}$-admissible modal substitutions in general.

7.2.2 $\mathcal{L}$-Admissible substitutions.

The existence of an $\mathcal{L}$-admissible substitution is an indication of the existence of a reduction order of the quantifiers and modal operators in the formula via sequent rules (when such calculi exist). We expect there to be only a finite number of correct reduction orders (derivations) for a given multiplicity. Consequently, we expect there to be only a finite number of distinct $\mathcal{L}$-admissible substitutions for a given multiplicity. We show that this is indeed the case.

Let A be a modal formula, X the signed modal formula $\langle A, 0\rangle$, μ be a multiplicity for X and σ an $\mathcal{L}$-admissible substitution for X^μ.

FACT 7.1 *$T_M(\mu)$ is finite.*

Let p be a prefix of the indexed formula tree for X^μ.

FACT 7.2 *For all $u \in \sigma(p)$, $u \in T_M(\mu)$.*

LEMMA 7.3 *$\sigma(p) \neq q_1 u q_2 u q_3$ for any $u \in T_M(\mu)$, and $q_i \in T_M(\mu)^*$, $i = 1, 2, 3$.*

PROOF. Suppose not. That is, suppose $\sigma(p) = q_1 u q_2 u q_3$. By Condition B on modal substitutions, we must have:

$$q_1 u = \sigma(\mathrm{pre}(u)) = q_1 u q_2 u,$$

which is absurd. ■

The above results indicate that there are a finite number of $\mathcal{L}$-admissible substitutions for a given multiplicity.

PROPOSITION 7.4 *The number of $\mathcal{L}$-admissible substitutions for X^μ is finite.*

PROOF. The subset of $T_M(\mu)^*$ comprising prefixes without repetitions is finite since $T_M(\mu)$ itself is finite. ■

7.2.3 The calculation of $\mathrm{Mgu}_M(U)$.

In this subsection we consider the calculation of $\mathrm{Mgu}_M(U)$ for each modal logic in turn. In what follows, if $p \in T_M(\mu)^*$, we use $\mathrm{len}(p)$ to denote the length of p (as a string).

7.2.3.1 K and D.

These logics, together with S5, are the simplest. The accessibility relation on prefixes for these logics is as follows (see Chapter 6):

$$p\ R_0\ q \quad \text{iff} \quad q = pu$$

for $u \in T_M(\mu)$ and $p, q \in T_M(\mu)^*$.

PROPOSITION 7.5 *If σ is a K or D-admissible substitution, for all $p \in T_M(\mu)^*$,*

$$len(\sigma(p)) = len(p)\,.$$

PROOF. Obvious. If this were not the case the substitution would not respect the accessibility relation. ■

Consequently, if we treat prefixes as strings of variables and constants, the standard unification algorithm suffices to compute most general substitutions that respect the K and D-accessibility relations. That is, variables are *not* considered string variables, and may only be instantiated to elements of $T_M(\mu)$.

We conclude that for these logics, the set $\mathrm{Mgu}_M(U)$ for a given set of connections in X^μ is a singleton set. Therefore $\mathrm{Mgu}(U)$ is a singleton set and the standard unification algorithm suffices for its calculation.

7.2.3.2 T.

The accessibility relation on prefixes for this logic is as follows (see Chapter 6):

$$p\ R_0\ q \quad \text{iff} \quad q = pu$$

for some $u \in T_M(\mu) \cup \{\emptyset\}$. That is, R_0 is reflexive. The appropriate length result for this logic is:

PROPOSITION 7.6 *If σ is a T-admissible substitution, for all $p \in T_M(\mu)^*$,*

$$len(\sigma(p)) \leq len(p)\,.$$

If we treat prefixes as strings of variables and constants, once again the standard unification algorithm suffices to compute most general substitutions that respect T-accessibility relations, provided we allow variables to "collapse" in the sense that they may be instantiated to the empty string/prefix. If instead of strings we introduce an explicit binary function symbol $*$ (*i.e.*, a prefix is a structure: $u_1 * (u_2 * (\cdots, u_n)) \cdots)$) then this unification problem can be shown to be an instance of unification under an assumption that $*$ is idempotent. That is, equational unification ([Sie84]) under the theory:

$$u * u \;=\; u.$$

For such unification problems, the set of most general unifers is finite, but not necessary a singleton [Sie84]. General algorithms are given in [Sie82].

We conclude that for T, Mgu(U) is not necessarily a singleton set but that algorithms exist for its calculation.

7.2.3.3 K4 and D4.

The accessibility relation on prefixes for these logics is as follows (see Chapter 6):

$$p \; R_0 \; q \quad \text{iff} \quad p \prec q$$

for $p, q \in T_M(\mu)^*$.

PROPOSITION 7.7 *If σ is a K4 or D4-admissible substitution, for all $p \in T_M(\mu)^*$,*

$$len(\sigma(p)) \geq len(p)\,.$$

Treating prefixes as strings once again, we must now consider the variables as true string variables that may be instantiated to *non-empty* strings only. Unification algorithms exist for general string unification, *eg.*, [Plo72,Sie75], but the set of most general unifiers of two strings p and q can be infinite [Sie84]. This situation arises because of the ability to repeat substrings and generate unifiers under which the image of p is of arbitrary length. The repetition constraint

(Lemma 7.3) restricts the full generality of the string unification. The set of most general unifiers is again finite but not necessarily a singleton set.

We conclude that for K4 and D4, $\mathrm{Mgu}_M(U)$ is finite.

7.2.3.4 S4.

The accessibility relation on prefixes for S4 is as follows (see Chapter 6):

$$p \, R_0 \, q \quad \text{iff} \quad p \preceq q$$

for $p, q \in T_M(\mu)^\star$. There is no length condition on the result of an S4-admissible substitution; it can be either longer or shorter. Once again, restricted string unification is the appropriate computational tool for computing most general unifiers of prefixes and hence most general S4-admissible substitutions. This time we admit the empty string as a valid instantiation of a variable.

The situation is reminiscent of unification under associativity and idempotence. Algorithms for this more general problem exist in the literature (*eg.*, [Sie82]).

We conclude once again that $\mathrm{Mgu}(U)$ is not necessarily a singleton set but that algorithms exist for its calculation.

7.2.3.5 S5.

The unification algorithm for computing most general modal substitutions for this logic is even more straightforward. The accessibility relation on prefixes for S5 is as follows (see Chapter 6):

$$u \, R_0 \, v \quad \text{iff} \quad u, v \in T_M(\mu).$$

That is, every unit prefix is accessible from every other. A similar length result holds for this logic as for K and D.

PROPOSITION 7.8 *If σ is an S5-admissible substitution, for all $u \in T_M(\mu)$,*

$$len(\sigma(u)) = len(u)\,.$$

Prefixes are simply variables or constants. The standard unification algorithm suffices to compute most general substitutions that respect the S5-accessibility relation.

We conclude that for S5, $\mathrm{Mgu}_M(U)$, and hence $\mathrm{Mgu}(U)$ is a singleton set. the standard unification algorithm suffices for its calculation.

7.2.4 Summary.

To summarise: we have argued that the only problematic component of the complementarity test in a modal logic is the calculation of most general modal substitutions that respect the $\mathcal{L}$-accessibility relation on prefixes for the logic. We have argued that in all cases this problem is a restriction of a more general problem for which suitable unification algorithms exist. In the case of K, D and S5, the standard unification algorithm suffices. If it exists, there is a unique most general (combined) $\mathcal{L}$-admissible substitution for a set of connections. For T, unification under an assumption of idempotency is suitable. There are a finite number of most general, T-admissible (combined) substitutions for a set of connections (when any exists at all). For the transitive logics K4, D4 and S4 associative, or string unification of prefixes is suitable. The general string unification problem may admit an infinite number of independent most general unifiers, but this only arises due to repetition. Since any such repetative substitution cannot be $\mathcal{L}$-admissible (Lemma 7.3) there are a finite number of most general, $\mathcal{L}$-admissible (combined) substitutions for a set of connections (when any exists at all). Modifying the string unification algorithm to eliminate substitutions inducing repetition is straightforward [Sie75].

7.3 Proof search in the matrix systems.

The matrix-based characterisations of validity presented in the previous chapter can be used to liberate validity checking in modal logics from the direct construction of a sequent proof tree. Instead, the validity of a sentence can be determined by showing that all the atomic paths through (an expansion) of the sentence viewed as a nested, two dimensional matrix, contain complementary connections. In the previous section we justified this view of matrix-based proof search by arguing that testing connections for complementarity is computationally tractable. The path checking task can be performed by first identifying a complementary connection within the sentence, and then eliminating from future consideration all atomic paths in which the connection appears as a subpath. If all the atomic paths through the sentence can be eliminated in this manner, it is valid. The basic search is confined to the decision as to which connections to make in the first place (cf.Part I). In this section we demonstrate that such matrix-based proof search is free from the redundancies demonstrated to arise in sequent-based proof search. In the next chapter we compare matrix-based proof search with the main resolution-based proposals in the literature for use with modal logics. Our conclusions are that the matrix characterisations developed above provide less redundant bases for automated proof search than these systems.

The redundancies identified with sequent-based search were classified under the following three headings:

- notational redundancy,
- relevance, and
- order dependence.

The reader is referred to Chapter 5 for a detailed discussion of these problems. We deal with each in turn.

7.3.1 Structure sharing and notational redundancy.

We noted in Chapter 5 that testing a modal formula for validity using a sequent/tableau calculus requires the construction of a proof tree/tableau for the appropriate endsequent. The search space is an OR-tree of possible derivations (partial proofs). Consequently, the use of any search strategy, except perhaps an incomplete depth-first search, requires the simultaneous storage of multiple derivations. Since a derivation itself is a tree consisting of many formulae, the space required to store such constructs can quickly become prohibitive.

A similar problem arises in resolution-based systems due to the number of resolvents that need to be stored during the use of standard resolution proof procedures. Boyer and Moore [BM72] devised a data-storage scheme which they termed *structure-sharing*. Descendants of this data-storage scheme are now standard components of resolution-based theorem proving systems. We shall not go into details of the structure-sharing technique since we discussed it in Part I. The basic idea is that new resolvents are not stored explicitly, but as a *skeleton* (consisting of pointers to the original set of clauses) together with a context marker that identifies the appropriate instantiations of the skeletal variables. In this way only the information necessary to construct the resolvent is retained, rather than the resolvent itself. Common structure is effectively *shared* between the intermediate constructs (in the case of resolution: clauses) manipulated by the proof procedure and the original set of clauses.

We showed in Part I that the formulation of such a technique relies, in essence, on the fact that classical logic admits (complete) proof systems that possess the *subformula property*: *i.e.*, any intermediate structure required in the proof of a formula within the proof system can be expressed entirely in terms of the subformulae of the original formula. The use of the notions of *formula tree*, *position* and *polarity* in the (theoretical) formulation of the matrix characterisation of validity for classical logic, were shown to directly support the implementation of matrix-based proof procedures that utilise structure-sharing methods. In short, positions can be interpreted as pointers to a single concrete representation of

the formula being tested for validity stored in a database. Each derivation, an intermediate state of the sequent search space, can be represented by pointers to the endsequent together with contextual information about the instantiations of free individual variables, instead of by explicit copies of formulae. Consequently, intermediate states in the matrix-based search space consist only of a collection of sets of pointers and context information, that allows the reconstruction of the leaves of derivations. The saving in space is considerable. The reader is referred to the arguments of Part I for more detail.

We noted in Chapter 5 that the modal sequent calculi also possess the subformula property. Consequently we adopted a similar method (also based on formula trees, positions and polarity) of capturing the applicability of structure-sharing in theorem provers based on the matrix characterisations of Chapter 6. Since the details are the same as the classical case we omit them here and refer the reader to the discussion of Part I. A structure-sharing scheme was used in this way in the implementation of a matrix-based theorem-prover for S5 [WW87].

7.3.2 Search strategies and relevance.

The problem that we have termed "relevance" within sequent-based proof search is basically a problem with the propositional fragment of the logics under consideration. Derivations are proofs just when their leaves are instances of the basic sequent; *i.e.,* contain a pair of propositionally complementary atomic formulae. The task of constructing proofs, as opposed to simply derivations, requires decisions as to which S-formulae to reduce to obtain such a pair in the resulting sequent. In Chapters $2\underset{\wedge}{\overset{\text{and}}{}}5$ we showed that *connective* driven search, typical of sequent-based proof systems, leads to larger search spaces than is strictly necessary since reductions are considered which cannot contribute to the construction of a basic sequent, and hence a proof. The switch to *connection* driven search overcomes these problems. The crucial point is that the connection search space is formulated in terms of the potential basic sequents that can be formed given the propositional structure of the formula we are trying to prove. The latter

structure is captured by the matrix representation of the formula; a representation that defines the set of atomic paths through the formula, and hence the potential leaves of a proof.

Since we rehearsed these arguments in detail in Part I we shall not repeat them here. Bibel [Bib82b,Bib82a] shows how some of the standard resolution search strategies can be utilised for this process. His results carry over to our modal systems without change. In addition, Bibel and his coworkers [Bib77, HB82,Bib82b] have investigated methods of improving the basic propositional component of matrix-based proof search. Their results transfer to the modal case without change since the propositional structure is the same.

7.3.3 Order dependence.

In the previous two subsections we briefly indicated how proof procedures based on the matrix ideas of Chapter 6 can be formulated so as not to suffer from the notational and relevance redundancies that plague sequent-based proof systems. The arguments were brief because the same structures had been used in Part I to overcome similar problems arising within classical logic. Due to the interactions of the rules for modalities and quantifiers, different reduction orders for the same set of modal operators and quantifiers will, in general, lead to essentially different derivations. Indeed, such derivations differ to the extent that it may be possible to construct a proof of the endsequent by extending one such derivation, but not from another. We have called this problem one of rule "order dependence." In the context of modal logic, this problem is perhaps the most important one to solve. We shall see in Chapter 8 that it is precisely this problem that most of the other proof systems for modal logics, based on resolution, and suggested as appropriate for automated proof search, *fail* to solve.

In this subsection we demonstrate how the matrix ideas of the previous chapter *do* overcome such problems. That is, we demonstrate that matrix-based proof search does not suffer from the redundancies associated with rule order depen-

dence. We do this by means of a series of examples designed to show the following features of matrix-based proof-search:

1. The first example shows how the (sequent-based) problem of choosing an appropriate reduction order for quantifiers and modal operators is overcome. The set of appropriate orders is *calculated* by unification rather than *explored* by search. The import of this fact is that the matrix-based search space is properly contained in the sequent-based search space.

2. A given order of reduction, appropriate at one point in the search, can be rendered inappropriate by a subsequent construction. If, as in sequent-based search, we are required to choose an explicit order in the first place, our choice may require subsequent revision. The second example shows how matrix-based search removes the need for such revision by avoiding overcommitment.

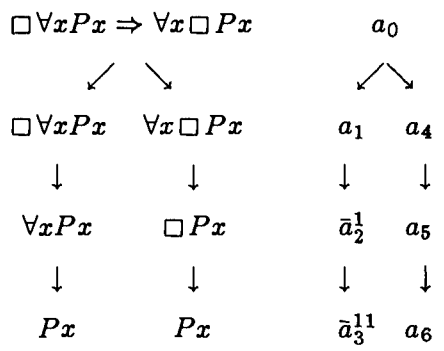
3. The third example demonstrates how extra copies of γ and ν-type formulae are considered *by need* in matrix-based search, rather than *arbitrarily* as is the case in sequent-based search. The theoretical treatment of the duplication of subformulae provided by the matrix characterisations also supports the formulation of efficient decision procedures for the propositional fragments of the modal logics. We explore this feature further in §7.4.

4. The matrix characterisations distinguish the K-logics from the D-logics only in the extra condition placed on the modal substitution in the definition of, say, K-admissibility. The fourth example illustrates the rôle of this restriction.

EXAMPLE. The first example is an instance of the converse Barcan formula:

$$\Box \forall x P x \Rightarrow \forall x \Box P x,$$

and is designed to show how the problem of choosing a reduction order for quantifiers and modal operators to obtain a proof is overcome using the matrix



u	pol(u)	lab(u)	pre(u)	pre$_{S5}$(u)	Ptype(u)	Stype(u)
a_0	0	$\Box\forall xPx \Rightarrow \forall x\Box Px$	a_0	a_0	α	π_0
a_1	1	$\Box\forall xPx$	a_0	a_0	ν	α_1
$\bar{a}_2^1$	1	$\forall xPx$	$a_0\bar{a}_2^1$	$\bar{a}_2^1$	γ	ν_0
$\bar{a}_3^{11}$	1	$P\bar{a}_3^{11}$	$a_0\bar{a}_2^1$	$\bar{a}_2^1$	$-$	γ_0
a_4	0	$\forall x\Box Px$	a_0	a_0	δ	α_2
a_5	0	$\Box Pa_5$	a_0	a_0	π	δ_0
a_6	0	Pa_5	a_0a_6	a_6	$-$	π_0

Figure 7–1: Indexed formula tree for: $\langle \Box\forall xPx \Rightarrow \forall x\Box Px, 0\rangle$.

characterisations. An indexed formula tree for this formula (signed 0), with (constant) multiplicity 1, is shown in Figure 7–1.

There are six derivations within the sequent system of, say, S4, that differ in the order in which the modal operators and quantifiers are reduced, causing the introduction of their immediate subformulae into the derivation. The possible orders of introduction for these immediate subformulae are:

$$\bar{a}_2^1\bar{a}_3^{11}a_5a_6 \qquad a_5\bar{a}_2^1\bar{a}_3^{11}a_6$$
$$\bar{a}_2^1a_5\bar{a}_3^{11}a_6 \qquad a_5\bar{a}_2^1a_6\bar{a}_3^{11}$$
$$\bar{a}_2^1a_5a_6\bar{a}_3^{11} \qquad a_5a_6\bar{a}_2^1\bar{a}_3^{11}.$$

Notice that since $\forall xPx$ is a subformula of $\Box\forall xPx$, $\bar{a}_2^1$ must always be introduced before $\bar{a}_3^1$. Similarly for the pair a_5 and a_6.

A sequent-based proof system would be required to choose from amongst this collection. For S4, only one of these reduction orders can lead to a proof with the current multiplicity. The successful order is:

$$a_5 a_6 \bar{a}_2^1 \bar{a}_3^{11},$$

giving the following S4-proof:

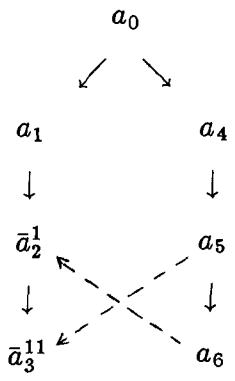
$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Pb \longrightarrow Pb}{\forall x Px \longrightarrow Pb}}{\Box \forall x Px \longrightarrow Pb}}{\Box \forall x Px \longrightarrow \Box Pb}}{\Box \forall x Px \longrightarrow \forall x \Box Px}}{\longrightarrow \Box \forall x Px \Rightarrow \forall x \Box Px}$$

All such considerations are overcome using the matrix characterisations. There is no search in this case since there is only one atomic path and one possible connection, namely: $\{\bar{a}_3^{11}, a_6\}$. The prefixes and labels of this connection are identified by the following (most general) combined substitution:

$$\begin{aligned} \sigma_M\left(\bar{a}_2^1\right) &= a_6 \\ \sigma_Q\left(\bar{a}_3^{11}\right) &= a_5. \end{aligned}$$

It is easy to check that this mapping is a modal substitution. Condition A holds trivially, since for no ν_0-type positions u and v do we have $\sigma_M(u) = v$. Condition B holds since $\mathrm{pre}(\bar{a}_2^1) = a_0 \bar{a}_2^1$ and $\mathrm{pre}(a_6) = a_0 a_6$, hence $\sigma_M(\mathrm{pre}(\bar{a}_2^1)) = \sigma_M(\mathrm{pre}(a_6))$.

As for $\mathcal{L}$-admissibility, the substitution respects all accessibility relations since only a unit sequence (a_6) is substituted for a variable. The reduction ordering (as a directed graph) for this substitution is:



which is acyclic. Notice that the graph represents the fact that a_6 must be introduced before $\bar{a}_2^1$, and hence, given the pairwise restrictions on $\bar{a}_2^1, \bar{a}_3^1$ and a_5, a_6, the only order of introduction of the six subformulae that leads to a proof is:

$$a_5 a_6 \bar{a}_2^1 \bar{a}_3^{11}.$$

The extra condition for K-logics is easily shown to be satisfied.

This argument shows that the substitution is admissible in the logics K, K4, D, D4, T, S4. Since there is only one path through the formula the connection spans the formula. (Note that for the K-logics the condition on complementarity is satisfied since all positions of the formula tree are in the set associated with the atomic path.) The sentence is therefore valid in the constant domain variants of these logics.

The first-order substitution has:

$$\sigma_Q\left(\bar{a}_3^{11}\right) = a_5,$$

where $\bar{a}_3^{11}$ is of ν_0-type and a_5 is of π_0-type. Furthermore,

$$\begin{aligned} \sigma_M\left(\mathrm{pre}\left(\bar{a}_3^{11}\right)\right) &= a_0 a_6 \\ \sigma_M(\mathrm{pre}(a_5)) &= a_0, \end{aligned}$$

and so

$$\sigma_M(\mathrm{pre}(a_5))\ R_0\ \sigma_M\left(\mathrm{pre}\left(\bar{a}_3^{11}\right)\right),$$

in the logics K, K4, D, D4, T, S4. This argument shows that the sentence is also valid in the cumulative domain variants of these logics.

The connection is *not* complementary in the varying domain systems, since:

$$\sigma_M\left(\mathrm{pre}\left(\bar{a}_3^{11}\right)\right) = a_0 a_6 \neq a_0 = \sigma_M(\mathrm{pre}(a_5)).$$

The non-complementarity of the atomic path for varying domains immediately gives us a two-point model, and a point in that model at which the formula fails to be forced. The general construction is as follows: we have two points,

a_0 and a_0a_6, with the latter accessible from the former. If u is a position with $\mathrm{pre}(u) = p$, then put:

$$\begin{cases} p \Vdash \mathrm{lab}(u)\,, & \text{if } \mathrm{pol}(u) = 1 \\ p \nVdash \mathrm{lab}(u)\,, & \text{if } \mathrm{pol}(u) = 0. \end{cases}$$

If u is a position of γ_0 or δ_0 type, with $\mathrm{pre}(u) = p$ put:

$$u \in \mathcal{D}(p)\,.$$

For the current example, these constructions produce:

w	$\mathcal{D}(w)$	$\{A \mid w \Vdash A\}$	$\{A \mid w \nVdash A\}$
a_0	$\{a_5\}$	$\Box\forall xPx$	$\forall x\Box Px, \Box Pa_5$
a_0a_6	$\{a_3^{11}\}$	$\forall xPx, Pa_3^{11}$	Pa_5

Notice that $a_0a_6 \Vdash \forall xPx$ and $a_0a_6 \nVdash Pa_5$. This is consistent because $a_5 \notin \mathcal{D}(a_0a_6)$. Notice also, that

$$a_0 \nVdash \Box\forall xPx \Rightarrow \forall x\Box Px$$

since $a_0 \Vdash \Box\forall xPx$ and $a_0 \nVdash \forall x\Box Px$. The reader can easily check that the above structure can be extended to an S4-model and hence is a falsifying model for all the other logics in their varying domain formulation except S5. (But it can be extended to one for S5 as well — see below.)

For S5, the most general combined substitution that identifies the prefixes and labels of the connection is:

$$\begin{aligned} \sigma_M\left(\bar{a}_2^1\right) &= a_6 \\ \sigma_Q\left(\bar{a}_3^{11}\right) &= a_5 \end{aligned}$$

i.e., identical to the combined substitution for the other logics. The reduction ordering is therefore acyclic and the sentence valid in constant domain S5. Since:

$$\sigma_M\left(\mathrm{pre}\left(\bar{a}_3^{11}\right)\right) = a_6 \neq a_0 = \sigma_M(\mathrm{pre}(a_5))\,,$$

the connection is *not* complementary in varying domain S5. The reader can easily check that the partial model given before can be extended to a falsifying, varying domain S5-model.

Notice how the sequent search amongst a large number of competing derivations is replaced by the deterministic use of unification and connections. The only choices that remain are: which connections to make, and what multiplicity to accept. The undecidability of the validity problem for the first order logics can be expressed in terms of the fact that we cannot determine in advance a maximum multiplicity within which we should restrict our search. See §7.4 for more details. (END OF EXAMPLE.)

EXAMPLE. The second example is the sentence:

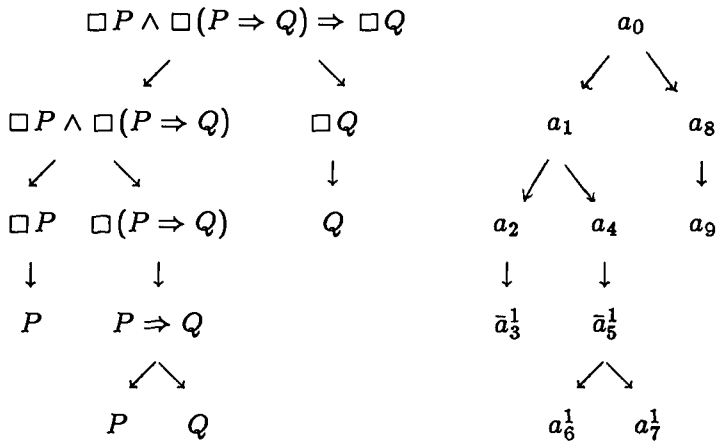
$$\Box P \wedge \Box(P \Rightarrow Q) \Rightarrow \Box Q.$$

This example is included to show how determining an appropriate order of reduction for modal operators (and quantifiers) cannot be done with regard to one connection in isolation. The decision is a global one to which each connection contributes. Within sequent-based search we are forced to choose an explicit order without knowing whether our particular choice will remain appropriate in the light of future constructions. In matrix-based search we postpone the choice. At each stage of the search enough information is maintained to check whether or not there is at least one correct derivation with the current set of connections at its leaves. The approriate condition is, of course, the irreflexivity of the reduction relation $\lhd$. The indexed formula tree, with (constant) multiplicity 1, is shown in Figure 7–2.

A T-derivation of this sentence is show below:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P \longrightarrow P, \Box Q} \quad \dfrac{\longrightarrow Q}{P, Q \longrightarrow \Box Q}}{P, P \Rightarrow Q \longrightarrow \Box Q}}{P, \Box(P \Rightarrow Q) \longrightarrow \Box Q}}{\Box P, \Box(P \Rightarrow Q) \longrightarrow \Box Q}}{\Box P \wedge \Box(P \Rightarrow Q) \longrightarrow \Box Q}}{\longrightarrow \Box P \wedge \Box(P \Rightarrow Q) \Rightarrow \Box Q}$$

There are six orders in which we can reduce the three distinct modal operators



u	pol(u)	lab(u)	pre(u)	pre$_{S5}$(u)	Ptype(u)	Stype(u)
a_0	0	$\Box P \wedge \Box(P \Rightarrow Q) \Rightarrow \Box Q$	a_0	a_0	α	π_0
a_1	1	$\Box P \wedge \Box(P \Rightarrow Q)$	a_0	a_0	α	α_1
a_2	1	$\Box P$	a_0	a_0	ν	α_1
$\bar{a}_3^1$	1	P	$a_0\bar{a}_3^1$	$\bar{a}_3^1$	–	ν_0
a_4	1	$\Box(P \Rightarrow Q)$	a_0	a_0	ν	α_2
$\bar{a}_5^1$	1	$P \Rightarrow Q$	$a_0\bar{a}_5^1$	$\bar{a}_5^1$	β	ν_0
a_6^1	0	P	$a_0\bar{a}_5^1$	$\bar{a}_5^1$	–	β_1
a_7^1	1	Q	$a_0\bar{a}_5^1$	$\bar{a}_5^1$	–	β_2
a_8	0	$\Box Q$	a_0	a_0	π	α_2
a_9	0	Q	a_0a_9	a_9	–	π_0

Figure 7–2: Indexed formula tree for: $\langle \Box P \wedge \Box(P \Rightarrow Q) \Rightarrow \Box Q, 0 \rangle$.

in the sentence and so introduce their immediate subformulae:

$$\bar{a}_3^1\bar{a}_5^1 a_9 \quad \bar{a}_5^1\bar{a}_3^1 a_9$$
$$a_9\bar{a}_3^1\bar{a}_5^1 \quad a_9\bar{a}_5^1\bar{a}_3^1$$
$$\bar{a}_5^1 a_9\bar{a}_3^1 \quad \bar{a}_3^1 a_9\bar{a}_5^1.$$

The derivation above corresponds to the order:

$$\bar{a}_3^1\bar{a}_5^1 a_9.$$

Notice that this order does not lead to a proof since one of the leaves of the derivation is not an instance of the basic sequent and there is no prospect of making it so (it contains only one atomic formula). So much for sequent-based search.

There are two atomic paths through the sentence with this multiplicity, and two possible connections within them. Consider the connection $\{\bar{a}_3^1, a_6^1\}$. The most general modal substitution necessary to identify the prefixes of this connection is:

$$\sigma_M\left(\bar{a}_3^1\right) = \bar{a}_5^1.$$

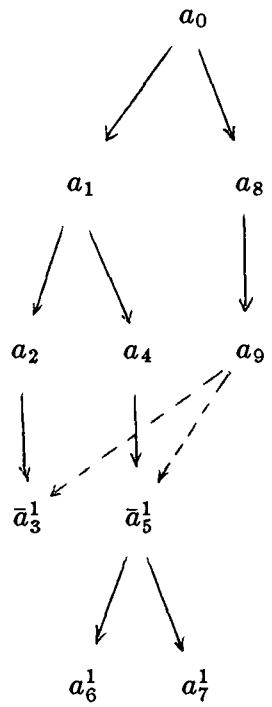
(Since the sentence is propositional, there is no need to consider a first-order substitution to identify the labels of the positions forming the connection; the labels are propositionally complementary by definition.)

The reduction ordering induced by this substitution is simply the tree ordering itself since the substitution only associates two "variables." The tree is obviously acyclic (as a directed graph). So far so good. This connection represents the left-hand leaf of the sequent derivation given above. The reduction order of that derivation: $\bar{a}_3^1\bar{a}_5^1 a_9$ is compatible with this connection since it is compatible with the formula tree.

The connection $\{\bar{a}_3^1, a_6^1\}$ spans only one of the atomic paths through the formula. The second is spanned by the other connection: $\{a_7^1, a_9\}$. The most general modal substitution that identifies the prefixes of both connections is:

$$\sigma_M\left(\bar{a}_3^1\right) = a_9$$
$$\sigma_M\left(\bar{a}_5^1\right) = a_9.$$

The reduction ordering induced by this substitution is:



which, once again, is acyclic. Notice that the reduction order we chose before, while appropriate for the first connection is inappropriate for the two connections in unison. (a_9 must be introduced before both $\bar{a}_3^1$ and $\bar{a}_5^1$.) A correct reduction order is therefore $a_0, a_8, a_9, \bar{a}_5^1$. Following this advice, we can construct a T-proof of the sentence:

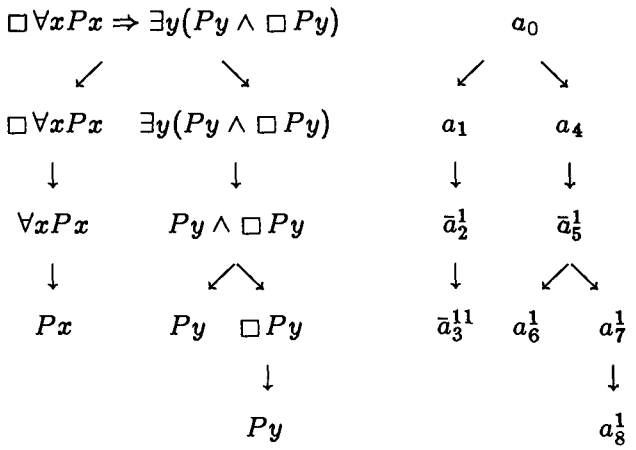
$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P \longrightarrow P, Q \qquad P, Q \longrightarrow Q}{P, P \Rightarrow Q \longrightarrow Q}}{\Box P, \Box(P \Rightarrow Q) \longrightarrow \Box Q}}{\Box P \wedge \Box(P \Rightarrow Q) \longrightarrow \Box Q}}{\longrightarrow \Box P \wedge \Box(P \Rightarrow Q) \Rightarrow \Box Q}$$

(END OF EXAMPLE.)

EXAMPLE. Our third example is the sentence:

$$\Box \forall x Px \Rightarrow \exists y(Py \wedge \Box Py).$$

This example is designed to show how extra copies of γ and ν-type formulae are considered *by need* in matrix-based search, rather than *arbitrarily* as is the case in sequent-based search. The indexed formula tree, with (constant) multiplicity 1, is shown in Figure 7–3.



u	pol(u)	lab(u)	pre(u)	pre$_{S5}$(u)	Ptype(u)	Stype(u)
a_0	0	$\Box\forall xPx \Rightarrow \exists y(Py \wedge \Box Py)$	a_0	a_0	α	π_0
a_1	1	$\Box\forall xPx$	a_0	a_0	ν	α_1
$\bar{a}_2^1$	1	$\forall xPx$	$a_0\bar{a}_2^1$	$\bar{a}_2^1$	γ	ν_0
$\bar{a}_3^{11}$	1	$P\bar{a}_3^{11}$	$a_0\bar{a}_2^1$	$\bar{a}_2^1$	–	γ_0
a_4	0	$\exists y(Py \wedge \Box Py)$	a_0	a_0	γ	α_2
$\bar{a}_5^1$	0	$P\bar{a}_5^1 \wedge \Box P\bar{a}_5^1$	a_0	a_0	β	γ_0
a_6^1	0	$P\bar{a}_5^1$	a_0	a_0	–	β_1
a_7^1	0	$\Box P\bar{a}_5^1$	a_0	a_0	π	β_2
a_8^1	0	$P\bar{a}_5^1$	$a_0a_8^1$	a_8^1	–	π_0

Figure 7–3: Indexed formula tree for: $\langle \Box\forall xPx \Rightarrow \exists y(Py \wedge \Box Py), 0\rangle$.

Two S4-derivations of this formula are shown below:

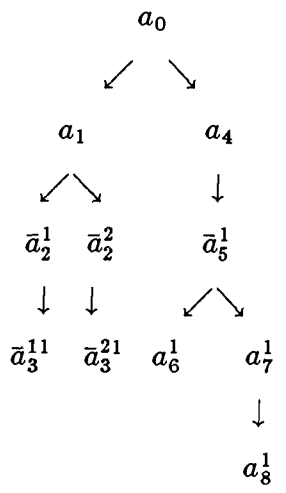
$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{Pb \longrightarrow Pb} \quad \dfrac{}{Pb \longrightarrow \Box Pb}}{Pb \longrightarrow Pb \wedge \Box Pb}}{Pb \longrightarrow \exists y(Py \wedge \Box Py)}}{\forall x Px \longrightarrow \exists y(Py \wedge \Box Py)}}{\Box \forall x Px \longrightarrow \exists y(Py \wedge \Box Py)}}{\longrightarrow \Box \forall x Px \Rightarrow \exists y(Py \wedge \Box Py)} \tag{7.1}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{Pb \longrightarrow Pb}}{\forall x Px \longrightarrow Pb}}{\Box \forall x Px \longrightarrow Pb}}{\Box \forall x Px,\ Pb \longrightarrow Pb} \quad \dfrac{}{\Box \forall x Px,\ Pb \longrightarrow \Box Pb}}{\Box \forall x Px,\ Pb \longrightarrow Pb \wedge \Box Pb}}{\Box \forall x Px,\ Pb \longrightarrow \exists y(Py \wedge \Box Py)}}{\Box \forall x Px,\ \forall x Px \longrightarrow \exists y(Py \wedge \Box Py)}}{\Box \forall x Px \longrightarrow \exists y(Py \wedge \Box Py)}}{\longrightarrow \Box \forall x Px \Rightarrow \exists y(Py \wedge \Box Py)} \tag{7.2}$$

The second is a proof. Notice how we are unable to close one of the leaves in the first derivation, while in the second this problem is overcome by retaining a "copy" of the antecedent formula $\Box \forall x Px$. This latter operation reflects the fact that we are working with *sets* of formulae. In general, within sequent-based search, we must be prepared to "copy" each formula of ν and γ type each time we generate an instance of its immediate subformula. In the proof we have only duplicated exactly that formula needed to close the leaf. In practice, many extra formulae must be retained in derivations so as to ensure completeness.

In matrix-based search the duplication of such formulae is governed by the requirements of connections. In other words, duplication is *demand-driven.* There are two atomic paths through the formula with this multiplicity. Likewise, there are two possible connections in the indexed formula of Figure 7–3 which together span it: $\{\bar{a}_3^{11}, a_6^1\}$ and $\{\bar{a}_3^{11}, a_8^1\}$. Consider the first connection. The most general combined substitution necessary to identify the prefixes and labels of this connection is:

$$\begin{aligned} \sigma_M\left(\bar{a}_2^1\right) &= \emptyset \\ \sigma_Q\left(\bar{a}_3^{11}\right) &= \bar{a}_5^1. \end{aligned}$$



u	pol(u)	lab(u)	pre(u)	pre$_{S5}$(u)	Ptype(u)	Stype(u)
$\bar{a}_2^2$	1	$\forall x P x$	$a_0\bar{a}_2^2$	$\bar{a}_2^2$	γ	ν_0
$\bar{a}_3^{21}$	1	$P\bar{a}_3^{21}$	$a_0\bar{a}_2^2$	$\bar{a}_2^2$	–	γ_0

Figure 7–4: Indexed formula tree with $\mu(\bar{a}_2) = 2$.

The reduction ordering induced by this substitution is simply the formula tree itself since the substitution only identifies variables.

Now consider the other connection $\{\bar{a}_3^{11}, a_8^1\}$. Under the current modal substitution we have:

$$\sigma_M\big(\mathrm{pre}\big(\bar{a}_3^{11}\big)\big) = a_0 \neq a_0 a_8^1 = \sigma_M\big(\mathrm{pre}\big(a_8^1\big)\big)\,,$$

hence the two connections cannot be made simultaneously complementary. This reflects the situation in the derivation 7.1 above. Since the failure to identify the prefixes is due to the modal substitution for $\bar{a}_2^1$, we increase the multiplicity of the lowest ν_0-type position dominating, or equal to $\bar{a}_2$. This is $\bar{a}_2$ itself. The new (indexed) formula tree for this multiplicity together with the information for the extra positions is shown in Figure 7–4.

Consider the connection $\{\bar{a}_2^2, a_8^1\}$. Under the current modal substitution we

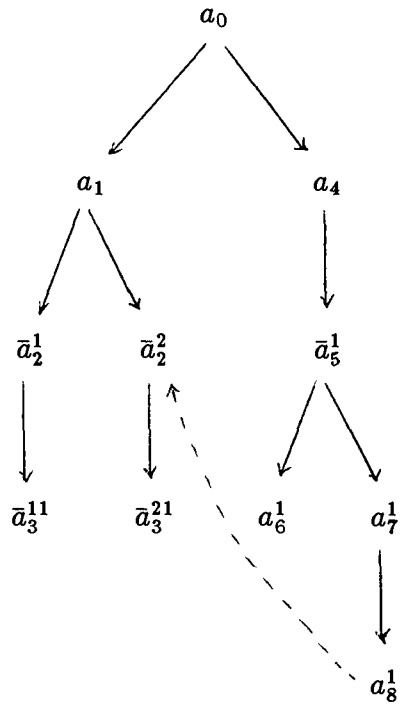
have:

$$\begin{aligned}\sigma_M\left(\mathrm{pre}\left(\bar{a}_2^2\right)\right) &= a_0\bar{a}_2^2\\ \sigma_M\left(\mathrm{pre}\left(a_8^1\right)\right) &= a_0a_8^1.\end{aligned}$$

The increase in multiplicity provides the necessary flexibility. These prefixes can be identified by extending the combined substitution with:

$$\begin{aligned}\sigma_M\left(\bar{a}_2^2\right) &= a_8^1\\ \sigma_Q\left(\bar{a}_3^{21}\right) &= \bar{a}_5^1.\end{aligned}$$

The reduction ordering remains acyclic:



The formula is therefore valid in the constant domain variants of the logics K, K4, D, D4, T, S4. Since

$$\begin{aligned}\sigma_M\left(\mathrm{pre}\left(\bar{a}_3^{11}\right)\right) &= a_0\\ \sigma_M\left(\mathrm{pre}\left(\bar{a}_3^{21}\right)\right) &= a_0a_8^1\\ \sigma_M\left(\mathrm{pre}\left(\bar{a}_5^1\right)\right) &= a_0\end{aligned}$$

the connections are also complementary in the cumulative domain variants, but not in the varying domain systems.

Notice how the increase in multiplicity was driven by the choice of connection. In sequent-based search, to retain completeness we must always perform such duplications. This increases the number of S-formulae that are candidates for reduction at any given stage and hence introduces redundant states in the search space. (END OF EXAMPLE.)

EXAMPLE. Our final example is simply to demonstrate the rôle of the extra conditions placed on the modal substitution in the definition of admissibility for the K-logics and when a connection ensures that an atomic path is complementary. These are the only conditions that distinguish the matrix characterisations for the K-logics from their D-counterparts.

Consider the formula:

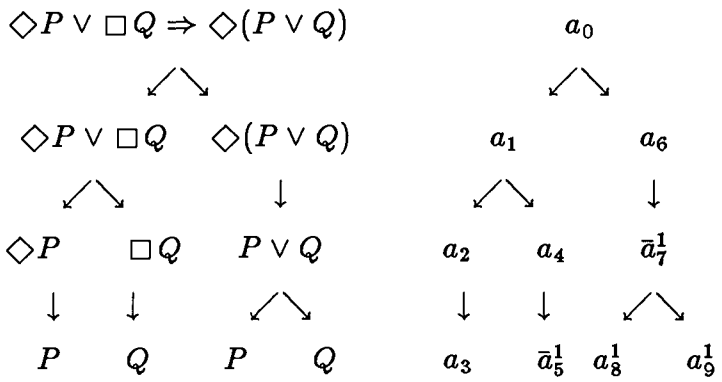
$$\Diamond P \vee \Box Q \Rightarrow \Diamond(P \vee Q).$$

An indexed formula tree for this formula, with (constant) multiplicity 1, is shown in Figure 7–5.

There are two atomic paths through the indexed formula, and two connections: $\{a_3, a_8^1\}$ and $\{\bar{a}_5^1, a_9^1\}$, which together span it. The prefixes and labels of these connections are identified by the substitution:

$$\begin{aligned} \sigma_M\left(\bar{a}_5^1\right) &= a_3 \\ \sigma_M\left(\bar{a}_7^1\right) &= a_3. \end{aligned}$$

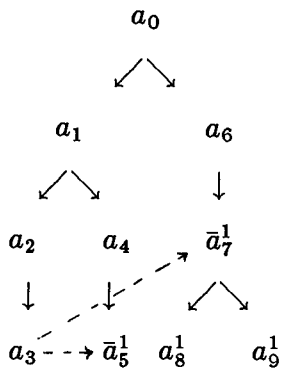
It is easy to see that this mapping is indeed a modal substitution, and admissible



u	pol(u)	lab(u)	pre(u)	pre$_{S5}$(u)	Ptype(u)	Stype(u)
a_0	0	$\Diamond P \vee \Box Q \Rightarrow \Diamond(P \vee Q)$	a_0	a_0	α	π_0
a_1	1	$\Diamond P \vee \Box Q$	a_0	a_0	β	α_1
a_2	1	$\Diamond P$	a_0	a_0	π	β_1
a_3	1	P	$a_0 a_3$	a_3	–	π_0
a_4	1	$\Box Q$	a_0	a_0	ν	β_2
$\bar{a}_5^1$	1	Q	$a_0 \bar{a}_5^1$	$\bar{a}_5^1$	–	ν_0
a_6	0	$\Diamond(P \vee Q)$	a_0	a_0	ν	α_2
$\bar{a}_7^1$	0	$P \vee Q$	$a_0 \bar{a}_7^1$	$\bar{a}_7^1$	α	ν_0
a_8^1	0	P	$a_0 \bar{a}_7^1$	$\bar{a}_7^1$	–	α_1
a_9^1	0	Q	$a_0 \bar{a}_7^1$	$\bar{a}_7^1$	–	α_2

Figure 7–5: Indexed formula tree for $\langle \Diamond P \vee \Box Q \Rightarrow \Diamond(P \vee Q), 0 \rangle$.

in D, D4, T, S4, S5. The reduction relation is shown below:



Informally, the K-condition comprises two components:

- the image, under the modal substitution, of a ν_0-type position in the prefix of an element of the connection must be comprised of π_0-type positions only; and

- these π_0 positions must be α-related to the ν_0 position.

The substitution above satisfies the first condition, but not the second since a_3 is not α-related to $\bar{a}_5^1$. The substitution is therefore not K or K4-admissible.

REMARK. Indeed there is no π_0-type position α-related to $\bar{a}_5^1$. Consequently the atomic positions with $\bar{a}_5^1$ in their prefix cannot take part in a K-complementary connection. This phenomenon is a subtle modal notion of *purity*, a concept familiar from resolution systems. A literal is taken to be pure if there is no complement for it within the set of clauses. The matrix counterpart of this notion is similar: an atomic position is pure if there is no complement for it in the matrix. Pure literals (atomic positions) cannot take part in a proof. The K-condition can thus be used to classify atomic positions as pure when attempting to determine the K-validity of formulae. (END OF REMARK.)

The K-condition can be motivated by looking at the structure of sequent derivations of our example formula. The reduction relation above tells us that

a_3 must be introduced before both $\bar{a}_5^1$ and $\bar{a}_7^1$. One such K-derivation is as follows:

$$\dfrac{\dfrac{\dfrac{P \longrightarrow P, Q}{P \longrightarrow P \vee Q}}{\Diamond P \longrightarrow \Diamond(P \vee Q)} \qquad \Box Q \longrightarrow \Diamond(P \vee Q)}{\dfrac{\Diamond P \vee \Box Q \longrightarrow \Diamond(P \vee Q)}{\longrightarrow \Diamond P \vee \Box Q \Rightarrow \Diamond(P \vee Q)}}$$

Recall that the sequent system for K has no ν rule, and the π rule is:

$$\frac{S*, \pi_0}{S, \pi} \quad \pi$$

where $S* \stackrel{\text{df}}{=} \{ \nu_0 \mid \nu \in S \}$. Hence the presence of the reduction:

$$\frac{P \longrightarrow P \vee Q}{\Diamond P \longrightarrow \Diamond(P \vee Q)}$$

in the derivation above. Moreover, *there is no rule applicable to the right-hand leaf sequent.* The K-condition represents the conditions under which such a dead-end will occur on the way to an instance of the basic sequent represented by a connection. In our case the connection is: $\{\bar{a}_5^1, a_9^1\}$.

As usual, we have enough information to construct a falsifying K-model for the formula from the non-complementary atomic path. Call this path s. Then,

$$s = \left\{ \bar{a}_5^1, a_8^1, a_4, a_6 \right\}$$

and

$$S(s) = \left\{ \bar{a}_5^1, a_8^1, a_4, a_6, a_0, a_1 \right\}.$$

The (partial) model contains only one point, a_0, at which the elements of $S(s)$ with prefix a_0 are forced, or not forced, according to their polarity (1 or 0 respectively):

w	$\{ A \mid w \Vdash A \}$	$\{ A \mid w \nVdash A \}$
a_0	$\Diamond P \vee \Box Q, \Box Q$	$\Diamond(P \vee Q)$

Basically $a_0 \Vdash \Box Q$ because there are no points accessible from a_0. Hence all such points force Q! There is no point accessible from a_0 at which $P \vee Q$ is forced. Consequently $a_0 \nVdash \Diamond(P \vee Q)$. Since a_0 forces the antecedent and fails to force the consequent of an implication, it also fails to force the implication itself. Notice that it is irrelevant which of $a_0 \Vdash Q$ or $a_0 \nVdash Q$ we take. (END OF EXAMPLE.)

7.4 Decision procedures.

In §7.3 we demonstrated how the duplication of certain subformulae (ν and γ-type subformulae) could be driven by the goal of making a given set of connections simultaneously complementary. Duplication is achieved by increasing the multiplicity. For a given multiplicity, the set of atomic paths through the indexed formula is finite. We can therefore search the resulting space exhaustively for a spanning set of connections and appropriate admissible substitution. Increasing the multiplicity, in general, increases the number of atomic paths through the indexed formula, thereby increasing the size of the search space. The matrix characterisations of validity, in terms of the existence of a multiplicity, suggests a modal analogue of Herbrand's Theorem.

For the quantified logics we cannot determine *in advance* a fixed multiplicity within which to search such that, if no substitution, and spanning set of complementary connections are found within that (finite) space, the formula is not valid. This is a reflection of the undecidability of the validity problem for these logics. On the other hand, the propositional fragments of the modal logics are decidable. This suggests that it is possible to determine from the structure of a formula, a "maximal" multiplicity within which to search for a proof of its validity. In this section we outline how the matrix characterisations provide a powerful framework for formulating decision procedures for these fragments by supporting the calculation of such maximal multiplicities.

The heart of the method is the notion of an *$\mathcal{L}$-Hintikka multiplicity* for a propositional formula A. An $\mathcal{L}$-Hintikka multiplicity, μ, has the property that if there is no $\mathcal{L}$-admissible modal substitution σ, and set of σ-complementary connections that spans $\langle A, 0\rangle^{\mu}$, then A is not valid. An $\mathcal{L}$-Hintikka multiplicity for a formula provides an effective bound on the size of the space to be searched by limiting the possibilities for duplication. If we can't prove the formula valid within these bounds, we are permitted to conclude that it is not valid.

In the next subsection we show how to determine a reasonably small S5-Hintikka multiplicity for propositional formulae. We concentrate on S5 for simplicity in order to illustrate the technique. At the time of writing S5 is the only logic for which an interesting and computationally significant construction has been defined and proved correct. In §7.4.2 we briefly indicate how to extend this result to the other logics.

REMARK. We stress: defining *an* $\mathcal{L}$-Hintikka multiplicity for a given formula for the other logics is straightforward, since the number of subformulae of the formula is finite. This, essentially, is the approach taken in tableau methods, such as Fitting's [Fit83], and Hughes and Cresswell's diagrammatic methods [HC68], to show that various modal logics are decidable in the first place. For the purposes of automated deduction, since the $\mathcal{L}$-Hintikka multiplicity defines the space that must be searched exhaustively to determine the non-validity of the formula, we are motivated by the desire to determine least such multiplicities. It is this problem that requires some more technical effort for a satisfactory resolution for the other logics. That being so, all the fundamental ideas necessary for such extensions are, we believe, contained in the S5 case below. The problem is simply one of time. (END OF REMARK.)

7.4.1 A decision procedure for S5.

In this subsection we formulate a decision procedure for S5 by giving a construction for an S5-Hintikka multiplicity for propositional modal formulae. The method is as follows:

- First we define the *modal degree* of a formula. Intuitively the degree of a formula indicates the maximum nesting of modal operators in the formula. Hughes and Cresswell [HC68] show how, for S5 at least, any formula may be transformed into an equivalent formula of first-degree.

- Next, we give a construction of a special multiplicity $\bar{\mu}$ for a given formula.

- Finally, we demonstrate that such multiplicities are $\mathcal{L}$-Hintikka multiplicities provided the formula is of at most *first-degree.*

In the next section we outline how the restriction to first-degree formulae can be lifted for S5, and the method extended to the other modal logics.

7.4.1.1 Modal degree.

The *modal degree* of a formula is defined inductively on the structure of formulae as follows ([HC68]):

1. Atomic formulae are of degree 0.

2. If A and B are formulae of degree n and m respectively, $\neg A$, $A \wedge B$, $A \Rightarrow B$ and $A \vee B$ are of degree $\max(n, m)$.

3. If A is a formula of degree n, $\Box A$ and $\Diamond A$ are of degree $n + 1$.

For example, if P and Q are atomic formulae, $\Box(P \wedge Q)$ is of degree 1, whereas $\Box(\Diamond P \wedge \Box\Diamond Q)$ is of degree 3. If the degree of a formula is 1, we say it is of *first-degree.*

The reader is referred to Hughes and Cresswell's book [HC68] for a procedure for transforming arbitrary propositional modal formulae to equivalent formulae of first-degree in S5. The transfomation basically rests on the S5-validity of equivalences of the form:

$$\begin{aligned} \Box(A \wedge B) &\equiv (\Box A \wedge \Box B) \\ \Diamond(A \vee B) &\equiv (\Diamond A \vee \Diamond B) \end{aligned}$$

$$\begin{aligned} \Box(A \vee \Box B) &\equiv (\Box A \vee \Box B) \\ \Box(A \vee \Diamond B) &\equiv (\Box A \vee \Diamond B) \end{aligned}$$

and

$$\begin{aligned} \Diamond(A \wedge \Box B) &\equiv (\Diamond A \wedge \Box B) \\ \Diamond(A \wedge \Diamond B) &\equiv (\Diamond A \wedge \Diamond B). \end{aligned}$$

7.4.1.2 The construction of $\bar{\mu}$.

Let A be a propositional modal formula and X the signed formula $\langle A, 0\rangle$. Recall that we use Π_0 to denote the set of unindexed positions of π_0-type in the unindexed formula tree. Recall also that, for a given multiplicity μ for X, $\Pi_0(\mu)$ and $\mathcal{V}_0(\mu)$ denote the set of π_0 and ν_0-type positions of the indexed formula X^μ respectively. As a final extension of this notation, if s is a path through X^μ, let $\mathcal{V}_0(\mu, s)$ denote those elements of $\mathcal{V}_0(\mu)$ that are elements of the set associated with the path s; similarly for $\Pi_0(\mu, s)$. Formally,

$$\begin{aligned} \mathcal{V}_0(\mu, s) &\stackrel{\text{df}}{=} \mathcal{V}_0(\mu) \cap \mathcal{S}(s) \\ \Pi_0(\mu, s) &\stackrel{\text{df}}{=} \Pi_0(\mu) \cap \mathcal{S}(s)\,. \end{aligned}$$

Define $M(k) \subseteq \Pi_0$ to be the set of π_0-type positions α-related to the position k of the unindexed formula tree for X. (Recall that two positions are α-related just in case they are both contained in some path through X.) Likewise, for a multiplicity μ and position k^κ of the indexed formula tree for X^μ, define $M(\mu, k^\kappa)$ to be the set of π_0-type positions α-related to k^κ in the indexed formula tree.

The multiplicity $\bar{\mu}$ is then defined as follows: for $k \in \mathcal{V}_0$,

$$\bar{\mu}(k) \;=\; |M(k)| + 1.$$

We claim that $\bar{\mu}$ is an S5-Hintikka multiplicity for A provided A is of at most first-degree. Before proving the claim, we illustrate the construction with an example.

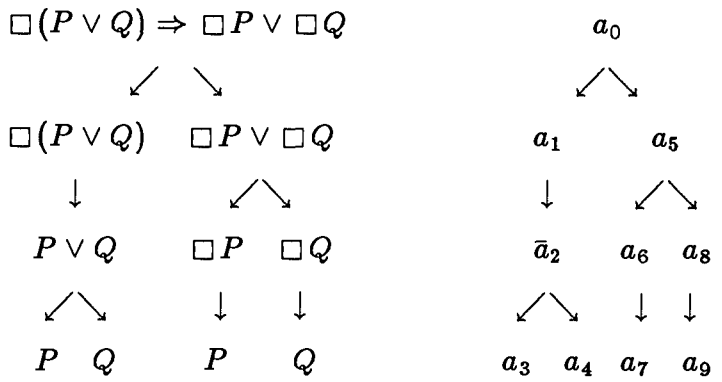
EXAMPLE. Consider the first-degree formula:

$$\Box(P \vee Q) \Rightarrow \Box P \vee \Box Q.$$

An unindexed formula tree for this formula is shown in Figure 7–6. Since $M(\bar{a}_2) = \{a_7, a_9\}$, $\bar{\mu}$ for this formula is the multiplicity:

$$\bar{\mu}(\bar{a}_2) = 3.$$

(END OF EXAMPLE.)



u	pol(u)	lab(u)	pre(u)	$\text{pre}_{S5}(u)$	Ptype(u)	Stype(u)
a_0	0	$\Box(P \vee Q) \Rightarrow \Box P \vee \Box Q$	–	a_0	α	π_0
a_1	1	$\Box(P \vee Q)$	–	a_0	ν	α_1
$\bar{a}_2$	1	$P \vee Q$	–	a_0	β	ν_0
a_3	1	P	–	$\bar{a}_2$	–	β_1
a_4	1	Q	–	$\bar{a}_2$	–	β_2
a_5	0	$\Box P \vee \Box Q$	–	a_0	α	α_2
a_6	0	$\Box P$	–	a_0	π	α_1
a_7	0	P	–	a_6	–	π_0
a_8	0	$\Box Q$	–	a_0	π	α_2
a_9	0	Q	–	a_8	–	π_0

Figure 7–6: Formula tree for $\langle \Box(P \vee Q) \Rightarrow \Box P \vee \Box Q, 0 \rangle$.

7.4.1.3 The proof.

In this section we work under a global assumption that A is at most a first-degree formula. We prove that the multiplicity $\bar{\mu}$ is an S5-Hintikka multiplicity for A.

First note some immediate consequences of the restriction to first-degree formulae. The lemmata indicate that increases in multiplicity do not increase the number of distinct π_0-type positions α-related to any position. It gives some indication as to why we defined $\bar{\mu}$ in terms of the unindexed formula tree (via $M(k)$), and why it suffices as a Hintikka multiplicity for first-degree formulae.

LEMMA 7.9 $\Pi_0(\mu) = \Pi_0$.

> PROOF. Since A is at most first-degree there are no nested modalities. An increase in multiplicity does not duplicate any π_0-type positions. ■

LEMMA 7.10 *For a multiplicity μ and indexed position k^κ of X^μ, $M(\mu, k^\kappa) = M(k)$.*

> PROOF. This is again an immediate consequence of the fact that A is at most first-degree. By the previous lemma, duplication does not increase the number of π_0-type positions in the indexed formula. ■

Next, we introduce some notions concerning multiplicities. Let $\mathbf{0}$ denote the constant zero multiplicity for X, and $\mathbf{1}$ the constant multiplicity equal to 1. We have the following fact: (recall that $\mathcal{A}(\mu)$ denotes the set of atomic paths through X^μ)

FACT 7.11 *For $s \in \mathcal{A}(\mathbf{0})$, if $u \in s$ either (a) u is an atomic position, or (b) $u \in \mathcal{V}$ (i.e., of ν-type).*

Let μ and μ' be multiplicities for X. μ' is said to be an *extension* of μ, written: $\mu' \geq \mu$, just in case for every $k \in \mathcal{V}_0$, $\mu'(k) \geq \mu(k)$. Notice that all multiplicities are extensions of $\mathbf{0}$.

Let s be a path through X^μ, for some multiplicity μ, and suppose $u \in \mathcal{S}(s)$. Define $s[u]$ to be the subpath of s through the subtree rooted at u. That is,

$$s[u] \;=\; \{\, v \in s \mid u \ll v \,\}.$$

We define an equality $\approx$ on sets of indexed positions by "forgetting" the indices. For a set of indexed positions, S, define basic(S) as follows:

$$\text{basic}(S) \;\stackrel{\text{df}}{=}\; \{\, k \mid k^\kappa \in S \,\}.$$

Let μ and μ' be two multiplicities for X, and s and s' two sets of indexed positions of X^μ and $X^{\mu'}$ respectively.

$$s \approx s' \;\text{ iff }\; \text{basic}(s) = \text{basic}(s').$$

Since A is of first-degree, any atomic path through X^μ can be decomposed into an atomic path through X^0 together with the subpaths through the individual ν_0-type position of $\mathcal{S}(s)$. Formally,

LEMMA 7.12 *Let μ be a multiplicity for X. Any $s \in \mathcal{A}(\mu)$ partitions into distinct sets: s_0, and $s[u]$ for $u \in \mathcal{V}_0(\mu, s)$, such that $s_0 \in \mathcal{A}(0)$ and $s[u]$ is a subpath of s with*

$$s \;=\; s_0 \cup \bigcup_{u \in \mathcal{V}_0(\mu,s)} s[u]$$

Furthermore, for any $s_0 \in \mathcal{A}(0)$ there is at least one path s through X^μ with s_0 as a subpath.

PROOF. Immediate from Fact 7.11 and the fact that A contains no nested modalities. ■

From these definitions and lemmata we get:

LEMMA 7.13 *For any $\mu \geq 0$, $\Pi_0(\mu, s) = \Pi_0(0, s_0)$, for some $s_0 \in \mathcal{A}(0)$ as described above.*

We prove the main claim in the following way. First we define a particular modal substitution $\bar{\sigma}$ over $X^{\bar{\mu}}$. This substitution essentially ensures that every possible distinct substitution is considered for a given ν_0-type position. ($\bar{\mu}$ is sufficiently large to allow this.) Then we demonstrate that if $\mathcal{A}(\bar{\mu})$ contains a non-complementary path (*i.e.*, there is no spanning set of $\bar{\sigma}$-complementary connections for $X^{\bar{\mu}}$), any extension $\mu \geq \bar{\mu}$ cannot alter this situation, for any modal substitution. We prove this latter part, by showing that $\mathcal{A}(\mu)$ contains an atomic path which is structurally identical to the non-complementary atomic path in $\mathcal{A}(\bar{\mu})$.

Suppose, for $k \in \mathcal{V}_0$,

$$M(k) = \{ r_1, r_2, \ldots, r_m \}.$$

Let k_0 denote the root position of the unindexed formula tree for A. Define the substitution $\bar{\sigma}$ as follows:

$$\bar{\sigma}(k^i) = \begin{cases} r_i, & 1 \leq i \leq m; \\ k_0, & i = m+1. \end{cases}$$

It is easy to check that $\bar{\sigma}$ is a modal substitution. Next, notice that $\bar{\sigma}$ is S5-admissible, since:

1. The mapping preserves S5-accessibility relations on prefixes since any prefix is accessible from any other.

2. The reduction relation induced by $\bar{\sigma}$ is irreflexive since, for no two distinct positions $u, v \in T_M(\bar{\mu})$ do we have $u \ll v$. (Recall that $T_M(\bar{\mu}) = \mathcal{V}_0(\bar{\mu}) \cup \Pi_0(\bar{\mu})$.)

As a final preliminary we prove a result concerning the multiplicity $\bar{\mu}$ and the substitution $\bar{\sigma}$ which expresses its "completeness" in a certain sense.

LEMMA 7.14 *Let $s \in \mathcal{A}(\bar{\mu})$ and $k \in basic(\mathcal{V}_0(\bar{\mu}, s))$. Then, for all π_0-type positions $v \in \Pi_0(\bar{\mu}, s)$ there is some index τ, with $1 \leq \tau \leq \bar{\mu}(k)$, such that $\bar{\sigma}(k^\tau) = v$.*

PROOF. Immediate, from the definition of $\bar{\sigma}$ and the fact that $\Pi_0(\bar{\mu}, s) \subseteq M(k)$. ■

PROPOSITION 7.15 *Let μ be an extension of $\bar{\mu}$, and σ any S5-admissible substitution for X^{μ}. If there is an atomic path $s \in \mathcal{A}(\bar{\mu})$ that does not contain a $\bar{\sigma}$-complementary connection, then there is a atomic path $t \in \mathcal{A}(\mu)$ that does not contain a σ-complementary connection.*

PROOF. By Lemma 7.12,

$$s = s_0 \cup \bigcup_{u \in \mathcal{V}_0(\bar{\mu}, s)} s[u]$$

for some atomic path $s_0 \in \mathcal{A}(0)$ and particular subpaths, $s[u]$, through $u \in \mathcal{V}_0(\bar{\mu}, s)$. We are free to choose an atomic path $t \in \mathcal{A}(\mu)$ such that:

$$t = s_0 \cup \bigcup_{u \in \mathcal{V}_0(\mu, t)} t[u]$$

for some subpaths $t[u]$, through $u \in \mathcal{V}_0(\mu, t)$. We choose the $t[u]$ as follows: let $u = l^{\kappa}$,

1. Suppose $\sigma(l^{\kappa}) = v$ for some π_0-type position v. By Lemma 7.13, $v \in \Pi_0(\bar{\mu}, s)$. By Lemma 7.14, there is an index τ, with $1 \leq \tau \leq \bar{\mu}(l)$, such that $\bar{\sigma}(l^{\tau}) = v$. Choose for $t[l^{\kappa}]$ the subpath through l^{κ} such that:

$$t[l^{\kappa}] \approx s[l^{\tau}].$$

2. Suppose $\sigma(l^{\kappa}) = v$ for some ν_0-type position v. By Lemma 7.13, there is some index τ, with $1 \leq \tau \leq \bar{\mu}(l)$, such that $\bar{\sigma}(l^{\tau}) = k_0$, where k_0 is the root position of the formula tree. (Recall $k_0 \in \Pi_0$ and $k_0 \in \mathcal{S}(s')$ for all paths s'. Consequently $k_0 \in \Pi_0(\bar{\mu}, s)$.) Choose for $t[l^{\kappa}]$ the subpath through l^{κ} such that:

$$t[l^{\kappa}] \approx s[l^{\tau}].$$

In both cases:

$$r^{\kappa} \in t[l^{\kappa}] \quad \text{iff} \quad r^{\tau} \in s[l^{\tau}].$$

(All the positions of $t[l^{\kappa}]$ have index κ since A is at most first-degree. Similarly for the positions of $s[l^{\tau}]$.) Hence, by construction:

$$t \approx s.$$

We claim that this path cannot contain a σ-complementary connection. Suppose not. That is, suppose there is some σ-complementary connection $\{k_1^{\kappa_1}, k_2^{\kappa_2}\}$ in t. Without loss of generality, suppose the prefixes of these atomic positions are $l_i^{\kappa_i}$, $i = 1, 2$, respectively. We have, by assumption:

- $\sigma(l_1^{\kappa_1}) = v = \sigma(l_2^{\kappa_2})$.
- $\mathrm{lab}(k_1^{\kappa_1}) = \mathrm{lab}(k_2^{\kappa_2})$.

Regardless of whether v is of π_0 or ν_0-type, we chose t such that there are some indices τ_1 and τ_2 such that:

$$\bar{\sigma}(l_i^{\tau_i}) = v', \qquad i = 1, 2,$$

for some π_0-type position $v' \in \Pi_0(\bar{\mu}, s)$, and

$$t[l_i^{\kappa_i}] \approx s[l_i^{\tau_i}] \qquad i = 1, 2.$$

That is, $k_i^{\tau_i} \in s$, $i = 1, 2$, and $\{k_1^{\tau_1}, k_2^{\tau_2}\}$ forms a $\bar{\sigma}$-complementary connection in s. This contradicts our assumption that no such connection exists. ■

THEOREM 7.16 *If A is a propositional modal formula of at most first-degree, $\bar{\mu}$ for A is an S5-Hintikka multiplicity.*

PROOF. We must show that if there is no $\mathcal{L}$-admissible modal substitution σ and set of σ-complementary connections that span $\langle A, 0\rangle^{\bar{\mu}}$, then A is not valid. Proposition 7.15 tells us that if $\bar{\sigma}$ is not sufficient to obtain a spanning set of $\bar{\sigma}$-complementary connections, no increase in multiplicity and alternative substitution is sufficient. By the completeness of the matrix characterisations, this means that A is not valid. ■

7.4.2 Extensions.

We remark at this stage how a decision procedure for first-degree S5 formulae can be defined based on the development above. Firstly, during standard path-checking proof search we place a bound on the multiplicity for each ν_0-type

position of the formula. Secondly, we need never consider substitutions σ that have the property:

$$\sigma(k^{\kappa_1}) = v = \sigma(k^{\kappa_2})$$

for some π_0-type position v; or

$$\sigma(k^{\kappa_1}) = \sigma(k^{\kappa_2}) = \sigma(k^{\kappa_3}),$$

for distinct κ_i, $i = 1, 2, 3$. These restrictions can be motivated by looking at the structure of the substitution $\bar{\sigma}$ defined above.

Recall that our goal is to define least such $\mathcal{L}$-Hintikka multiplicities. We can improve on $\bar{\mu}$ defined above by taking account of the fact that the identification of two distinct positions under the substitution can only produce complementary connections in the formula if there are (propositionally complementary) atomic formulae with those positions as prefixes. For example, consider this variant of the example formula of the previous section:

$$\Box(P \vee Q) \Rightarrow (\Diamond P \Rightarrow \Box Q)$$

(The formula tree for this formula is isomorphic to that shown in Figure 7–6, right down to the types of the positions. The only difference is that the two occurrences of P have the same polarity. We take advantage of this and use the same names to refer to the corresponding positions.) The construction given above would still produce an S5-Hintikka multiplicity $\bar{\mu}$ with

$$\bar{\mu}(\bar{a}_2) = 3.$$

We can show that the smaller multiplicity:

$$\bar{\mu}(\bar{a}_2) = 1$$

is also an S5-Hintikka multiplicity and clearly leads to a smaller search space containing one possible connection only. The existence of a non-complementary path at this multiplicity immediately leads to the conclusion that the formula is not valid. Basically, instead of using all of the distinct positions α-related to the ν_0-type position, we only count those which prefix potential connections.

The construction and proof of Theorem 7.16 relies on the fact that an increase in multiplicity, in a manner of speaking, does not increase the number of distinct prefixes in the formula. Of course it does increase the number of ν_0-type positions of the unindexed formula tree, but we need only consider substitutions that map all ν_0-type positions to distinct π_0-type positions. (This is also reflected in the completeness proofs of Chapter 6, where the substitutions used to construct σ-complete atomic paths were "ground" in this sense.) The crucial lemma is Lemma 7.13. In fact, it is not difficult to show that very slight modifications of the S5-construction of the last section suffice for the other logics also, under the restriction to first-degree formulae of course.

To lift the restriction to formulae in first-degree normal-form for S5, and hence extend the method to the other logics, we need to calculate the increase in distinct prefixes (π_0-type positions) that occurs when the multiplicity is increased. Clearly this depends on the degree of the particular formula. Notice that for first-degree formulae we were able to define $\bar{\mu}$ in terms of $M(k)$ by

$$\bar{\mu}(k) \;=\; |M(k)| + 1.$$

The more general case requires the definition to be in terms of $M(\bar{\mu}, k^\kappa)$. The recursive nature of such a definition reflects the fact that an increase in the multiplicity for one position will, in general, force increases for other positions. We have not developed such characterisations to-date, but believe it to be a simple, if tedious technical exercise.

7.5 Logical consequence and expressibility.

In this section we consider two miscellaneous but important issues: namely, the use of the matrix characterisations to decide instances of the consequence relation for the modal logics, and the extension of the characterisations to languages containing function symbols.

7.5.1 Logical consequence

The matrix characterisations capture the structure of valid sentences within modal logics. In practice we are interested in the relation of *logical consequence* between sentences and sets of sentences. The characterisations suffice to decide instances of logical consequence.

The consequence relation for modal logics that we wish to capture is the following: the sentence A is a logical consequence of (the set of sentences) Θ just in case, for any $\mathcal{L}$-model, and any $\mathcal{L}$-interpretation ι in that model, for all $w \in G$,

$$w \Vdash \iota(\Theta) \quad \text{implies} \quad w \Vdash \iota(A).$$

We write $\Theta \models A$. Deduction theorems hold for the modal logics under the above notion of logical consequence.

THEOREM 7.17 (DEDUCTION THEOREM) *If A is a sentence, and Θ a set of sentences, then for all the modal logics under consideration:*

$$\Theta \models A \quad \textit{iff} \quad \models \Theta \Rightarrow A.$$

We can therefore use the matrix characterisations to decide instances of logical consequence by proving the validity of the implication $\Theta \Rightarrow A$, instead of the consequence $\Theta \models A$. (In implementations of course the distinction can be hidden. See [WW87].)

7.5.2 Function symbols.

We have considered languages containing no function symbols. This restriction was made for technical convenience. Function symbols can be added to the matrix systems in the same way as Bibel adds function symbols to his classical matrix system [Bib82a]. The only change occurs in the method for calculating the relation $\sqsubset_Q$ induced by a first-order substitution. The definition given was: a first-order substitution $\sigma_Q : \Gamma_0(\mu) \to T_Q(\mu) \cup C$ induces an equivalence relation $\sim_Q$ and a relation $\sqsubset_Q$ on $T_Q \times T_Q$ as follows:

1. If $\sigma_Q(u) = v$ and $v \in \Gamma_0$, then $u \sim_Q v$.

2. If $\sigma_Q(u) = v$ and $v \notin \Gamma_0$, then $v \sqsubset_Q u$.

3. If $v \sqsubset_Q u$ and $u \sim_Q u'$, then $v \sqsubset_Q u'$.

Since the image of a ν_0-type position under σ_Q can now be a term t, we alter the second clause to:

2. If $\sigma_Q(u) = t$ for some term t, for all $v \in T_Q(\mu)$ that are subterms of t, $v \sqsubset_Q u$.

Everything else goes through unchanged.

7.6 Summary.

In this chapter we have:

1. Justified the view of matrix-based proof search in modal logics as being essentially a path checking process (§7.2). We did this by demonstrating that the complementarity tests, as in classical logic, are computationally tractable. Indeed, algorithms already exist for computing most general $\mathcal{L}$-admissible substitutions.

2. Demonstrated that proof search based on the matrix characterisations overcomes the problems identified with sequent-based proof search (§7.3).

3. Shown how the characterisations support the development of efficient decision procedures for the propositional fragments of modal logics (§7.4).

4. Shown how the characterisations can be used to decide instances of modal consequence relations and the restriction to first-order languages with no function symbols can be trivially lifted (§7.5).

Chapter 8

Related work.

8.1 Introduction.

In the previous chapters we developed matrix characterisations of validity for the first-order modal logics K, K4, D, D4, T, S4 and S5. We captured not only the standard quantified logics, but also their cumulative and constant domain variants. We have succeeded in turning the task of checking a modal formula for validity into a path checking task, with each primitive operation a test for complementarity. We showed in the previous chapter that the complementarity tests are tractable and instances of standard (string) unification problems.

Our goal has been to extend techniques for automated deduction in classical logic to modal logics while retaining the computational properties of these techniques. We have succeeded in this goal. Indeed, search strategies based on the classical matrix systems are applicable *without change* in the modal case. The specifically "modal" aspects of the proof search are dealt with by unification during the complementarity tests. In [Bib82b], Bibel develops a series of path checking algorithms for classical matrices that are equivalent to the most efficient resolution strategies. His results suffice to demonstrate the degree of efficiency we have achieved for automated deduction in modal logics.

In this chapter we review the other major proposals in the literature for (computationally) efficient proof procedures for (roughly) the same class of modal logics. We identify three types of system:

- Proof systems based on sequent/tableau calculi (§8.2).
- Systems based on either clausal or non-clausal resolution (§8.3).
- Hybrid proof systems based on a mixture of sequent/tableau ideas and resolution (§8.4).

As discussed in the introduction to this thesis, comparing the efficiency of proof procedures is a difficult task. We restrict ourselves to investigating properties of the search spaces generated by the proposed inference systems. In some cases, such as the sequent/tableau systems, we can show that the matrix search space is a subspace of the space generated by the sequent/tableau system. In others, such as Abadi and Manna's adaptation of non-clausal resolution to modal logics [AM86a], we indicate the combinatorial problems possessed by the systems that make them much less appropriate for automated proof search than the matrix systems.

We note at the outset that, in the author's opinion, each of the proposals reviewed provide a more redundant basis for automated proof search than our modal matrix characterisations. The reason is simple. None of the proposals overcome what we have termed the "order dependence" of the modal rules, or more abstractly, the interaction of modalities. We point to our use of unification to solve this problem as a central contribution. Indeed, our threefold classification of the redundancies of sequent-based proof search can be used to summarise the problems with the other proposals:

- The spaces generated by sequent/tableau proof systems, in general contain all three types of redundancy: notational, relevance and order dependence.
- The spaces generated by the resolution and hybrid tableau/resolution proof systems:

- can be extended to utilise structure sharing methods to overcome the notational problems;
- overcome the relevance problem by means of connections,
- overcome the order problems of the quantifiers, but
- *fail* to deal with the order problems associated with modalities.

Additionally, none of the resolution proposals capture the full range of modal logics treated here, though in some cases the systems can be extended.

8.2 Sequent and tableau-based proof systems

In Chapter 5 we identified redundancies in the search spaces generated by standard sequent and tableau calculi for modal logics that render them inappropriate for automated proof search. These redundancies are common to all modal sequent/tableau systems with a characteristic emphasis on connectives and interacting inference rules. Such systems include those presented by Kripke [Kri63], Fitting [Fit83], Hughes and Cresswell [HC68], and the decision methods of Halpern and Moses [HM84] for various extended classes of modal logic. In Chapter 7 we demonstrated that proof search based on the matrix characterisations were free from the redundancies, and hence form more suitable bases for automated deduction in modal logics. We shall not repeat the arguments here, but refer the reader to the chapters cited.

Fitting's *prefixed tableau* systems [Fit72,Fit83] deserve further mention because they are related to the methods employed in the matrix characterisations. Indeed, our methods were partially inspired by his systems. For instance, a prefixed sequent system for S5, developed by Kanger [Kan57], was used to suggest a matrix characterisation for that logic in [Wal86]. In [Wal87] we used Fitting's prefixed systems to motivate the design of matrix characterisations, much as we used more standard sequent systems in this thesis.

$$\Gamma, p : A \longrightarrow p : A, \Delta$$

$$\frac{\Gamma, p : A, p : B \longrightarrow \Delta}{\Gamma, p : (A \wedge B) \longrightarrow \Delta} \wedge \longrightarrow \qquad \frac{\Gamma \longrightarrow p : A, \Delta \quad \Gamma \longrightarrow p : B, \Delta}{\Gamma \longrightarrow p : (A \wedge B), \Delta} \longrightarrow \wedge$$

$$\frac{\Gamma, p : A \longrightarrow \Delta \quad \Gamma, p : B \longrightarrow \Delta}{\Gamma, p : (A \vee B) \longrightarrow \Delta} \vee \longrightarrow \qquad \frac{\Gamma \longrightarrow p : A, p : B, \Delta}{\Gamma \longrightarrow p : (A \vee B), \Delta} \longrightarrow \vee$$

$$\frac{\Gamma \longrightarrow p : A, \Delta \quad \Gamma, p : B \longrightarrow \Delta}{\Gamma, p : (A \Rightarrow B) \longrightarrow \Delta} \Rightarrow \longrightarrow \qquad \frac{\Gamma, p : A \longrightarrow p : B, \Delta}{\Gamma \longrightarrow p : (A \Rightarrow B), \Delta} \longrightarrow \Rightarrow$$

$$\frac{\Gamma, p : A \longrightarrow \Delta}{\Gamma, p : (\neg A) \longrightarrow \Delta} \neg \longrightarrow \qquad \frac{\Gamma \longrightarrow p : A, \Delta}{\Gamma \longrightarrow p : (\neg A), \Delta} \longrightarrow \neg$$

$$\frac{\Gamma, q : A \longrightarrow \Delta}{\Gamma, p : (\Box A) \longrightarrow \Delta} \Box \longrightarrow \qquad \frac{\Gamma \longrightarrow q : A, \Delta}{\Gamma \longrightarrow p : (\Box A), \Delta} \longrightarrow \Box$$

$$\frac{\Gamma, q : A \longrightarrow \Delta}{\Gamma, p : (\Diamond A) \longrightarrow \Delta} \Diamond \longrightarrow \qquad \frac{\Gamma \longrightarrow q : A, \Delta}{\Gamma \longrightarrow p : (\Diamond A), \Delta} \longrightarrow \Diamond$$

- In all the modal rules we must have: $p \, R_0 \, q$.
- For the $\longrightarrow \Box$ and $\Diamond \longrightarrow$ rules: q must not appear in the conclusion.

Figure 8–1: Prefixed sequent calculus for S5.

Let $\langle G_0, R_0 \rangle$ be an $\mathcal{L}$-frame. A *prefixed formula* is a pair $p : A$, where $p \in G_0$ and A is a modal formula. The rules for sequent versions of Fitting's prefixed systems are shown in Figure 8–1. Fitting utilises sequences of integers for the set of prefixes G_0 and defines the accessibility relation R_0 in exactly the same way as we did in Chapter 6 for prefixes consisting of sequences of positions. Indeed we took the definitions from his [Fit83], as remarked at the time. For example, the accessibility relation for S4 is defined as:

$$p \, R_0 \, q \quad \text{iff} \quad p \preceq q.$$

The rules of the calculus are inverted for proof search, derivations being constructed from their root to their leaves. The use of prefixes allows Fitting, as it did us, to capture the constant domain logics easily. Despite the use of

prefixes however, *all three* types of redundancy remain in the prefixed sequent systems. They possess the notational and relevance redundancies by virtue of the standard sequent/tableau framework. Moreover, the proviso on the "π" rules: $\longrightarrow \Box$ and $\Diamond \longrightarrow$, induce interactions between these rules and the "ν" rules: $\Box \longrightarrow$ and $\longrightarrow \Diamond$ — classic order dependence. A simple example will help to illustrate this. Consider the S4-theorem: $\Box A \wedge \Box(A \Rightarrow B) \Rightarrow \Box A$. We begin by prefixing the formula with the prefix 1, and applying the rules for implication, conjunction and the ν rule $\Box \longrightarrow$ twice, thereby introducing the prefix 11, which has $1 \preceq 11$. (Note: we use 11 to denote the two element prefix formed from the concatenation of the two unit sequences: 1 and 1.) The resulting derivation is shown below.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{11 : A,\ 11 : (A \Rightarrow B) \longrightarrow 1 : (\Box A)}{11 : A,\ 1 : \Box(A \Rightarrow B) \longrightarrow 1 : (\Box A)}}{1 : (\Box A),\ 1 : \Box(A \Rightarrow B) \longrightarrow 1 : (\Box A)}}{1 : (\Box A \wedge \Box(A \Rightarrow B)) \longrightarrow 1 : (\Box A)}}{\longrightarrow 1 : (\Box A \wedge \Box(A \Rightarrow B) \Rightarrow \Box A)}$$

But now when we come to reduce the succedent formula, we are prohibited from introducing the prefix 11 by the proviso on the $\longrightarrow \Box$ rule. Of course, we should have reduced the succedent formula before the antecedent formulae.

In conclusion, despite the use of prefixes, Fitting's systems suffer from the same problems as the more standard sequent/tableau systems. It is possible to introduce our unification solution to the prefix systems directly. In fact, the motivational arguments used to introduce the matrix systems in Chapter 6 essentially did this. Jackson and Reichgelt [JR87] investigate this method further.

8.3 Resolution-based proof systems.

Perhaps the most obvious method of developing computationally efficient proof methods for modal logics is to adapt Robinson's resolution method [Rob65], and its refinements (see *eg.*, [CL73]). The idea would be to extend the resolution method to modal logics while retaining the computational advantages exhibited by the method for classical logic.

The major hurdle to this approach is that the standard resolution method requires that the input formula be in *clausal form.* The precise details of this normal form are not important. We refer the reader to Chang and Lee's book [CL73] for a full description. Suffice it to say the method relies on the validity of equivalences such as:

$$\forall x \Diamond A \equiv \Diamond \forall x A,$$

as well as the existence of a conjunctive normal form for the propositional fragments. We have already seen that modal operators and quantifiers do not commute freely. (In fact, the equivalence above is not valid in any of the logics we have considered, since the left-to-right implication fails.) Moreover, a simple conjunctive normal form does not exist for all the modal logics. (Hughes and Cresswell, in [HC68], define a modal clausal form for S5 and point out that its existence is dependent on the number of distinct modal functions, of a given set of propositional variables, expressible in the logic. For S5 this number is finite because, as we showed in Chapter 7, every S5 formula can be reduced to a formula of a particular maximum degree, namely 1. For S4 and T, there is no such finite degree to which formulae can be reduced. Hence there are an infinite number of distinct modal functions for these logics, and no such normal form exists.)

Consequently, a resolution approach must either restrict the language of formulae to, say, *prenex* formulae, where no quantifiers are allowed in the scope of modal operators, and abandon classical clausal form; or alternatively, abandon normal form altogther. The most comprehensive treatment of modal logics using

the former approach is that of Fariñas del Cerro [Far82,Far83,Far86]. The most comprehensive treatment of modal logics using the latter approach is that of Abadi and Manna [AM86b,AM86a]. We review both approaches below.

8.3.1 Clausal resolution for modal logics.

Fariñas del Cerro has developed clausal resolution systems for the propositional modal logics K [Far82], S4 and S5; and a linear temporal logic of programs [Far83]. In [Far86] he argues that the systems extend to the prenex subset of the first-order logics.

We note that these methods have not been developed for the wide class of propositional logics treated in this thesis. Secondly, we note that the extension of the systems to first-order modal logics requires the aforementioned syntactic restriction to prenex normal form; a restriction not shared by the matrix characterisations. Thirdly, Bibel's results in [Bib82b] suffice to show that search methods based on matrix characterisations are more efficient than standard refinements of classical clausal resolution. These results generalise immediately to the modal case since we were careful to preserve the path checking nature of the search. (Basically, Bibel interprets resolution inferences within a matrix framework and shows that each standard resolution inference fails to eliminate all the atomic paths that are proved complementary by the inference. In this way, more resolution inferences are performed than is strictly necessary. Part of this redundancy comes directly from the use of clausal form.)

Fariñas del Cerro's modal resolution inference rules are defined in the following manner. First, a modal "clausal form" is defined in which modal operators may quantify subclauses. For example, the formula:

$$\Box(P \vee Q \vee \Diamond(R \wedge T))$$

is in normal form because it is a conjunction of clauses (one in this case) each of which is of the form:

$$C = L_1 \vee \cdots \vee L_m \vee \Box C_1 \vee \cdots \vee \Box C_n \vee \Diamond D_1 \vee \cdots \vee \Diamond D_p,$$

where the C_j and D_k are themselves clauses, and the L_i literals. We shall briefly present the system for S5.

A single resolution inference is performed by means of a procedure on clauses. Write $\langle C_1, C_2 \rangle$ for clauses C_1 and C_2 are resolvable, and define this relation and the resolvent: $\Sigma(C_1, C_2)$, recursively as follows:

1. Classical rules:

   (a) For atomic P, $\Sigma(P, \neg P) = \emptyset$. $\langle P, \neg P \rangle$ is resolvable.

   (b) $\Sigma((D_1 \vee D_2), C) = \Sigma(D_1, C) \vee D_2$. If $\langle D_1, C \rangle$ is resolvable then so is $\langle D_1 \vee D_2, C \rangle$.

   (c) $\Sigma(D_1 \wedge C_1 \wedge D_2 \wedge C_2) = \Sigma(D_1 \wedge D_2) \wedge C_1 \wedge C_2$. If the clause $D_1 \wedge D_2$ is resolvable, so is the clause $D_1 \wedge C_1 \wedge D_2 \wedge C_2$.

2. Modal rules:

   (a) $\Sigma(\Box D, \Box C) = \Box \Sigma(D, C)$. If $\langle D, C \rangle$ is resolvable, so is $\langle \Box D, \Box C \rangle$.

   (b) $\Sigma(\Box D, \Diamond C) = \Diamond(\Sigma(D, C) \wedge C)$. If $\langle D, C \rangle$ is resolvable, then so is $\langle \Box D, \Diamond C \rangle$.

   (c) $\Sigma(\Box D, C) = \Sigma(D, C)$. If $\langle D, C \rangle$ is resolvable, so is $\langle \Box D, C \rangle$.

   (d) $\Sigma(B[\Diamond(D_1 \wedge D_2 \wedge C)]) = B[\Diamond(\Sigma(D_1, D_2) \wedge C \wedge D_1 \wedge D_2)]$. If $\langle D_1, D_2 \rangle$ is resolvable, so is $B[\Diamond(D_1 \wedge D_2 \wedge C)]$.

(The last rule of each class defines what it means for a clause to be resolvable with itself.)

Given two clauses, the above set of rules is run as a procedure to determine whether the clauses are resolvable and compute the resolvent. Notice that the procedure is non-deterministic. Choices must be made in two of the classical rules and the final modal rule. The resolution rule is therefore not *effective* in that a search must be undertaken to determine the results of the inference. In the case of the quantified logics this search will be arbitrary. The modal rules effectively search for classical inconsistencies (connections) within modal

contexts. This is no more than a particular strategy for applying tableau or sequent rules to produce an instance of the basic sequent. In fact it is worse, since the results of such applications to break down complex formulae are not stored in a sequent, or on a tableau, but are computed anew for each attempted resolution.

This behaviour should be compared with the method of identifying complementary connections in systems based on the matrix characterisations. In the matrix systems no normal forming is necessary, with its attendant increase in redundancy due to the expansion of the formula. We conclude that the clausal resolution systems are less suitable for automated proof search in modal logics than the modal matrix systems.

8.3.2 Non-clausal resolution for modal logics.

Abadi and Manna [AM86b,AM86a] develop resolution methods only for the constant domain formulations of the modal logics we have considered. It is not immediately obvious whether their methods extend to varying and cumulative domain variants. Their methods are based on Murray [Mur82] and Manna and Waldinger's [MW80] *non-clausal* resolution rule.

A resolution proof system for a given (constant domain) logic contains two types of rule:

1. Simplification rules.

2. Deduction rules:

   (a) The resolution rule.

   (b) Modal rules.

Simplification rules have the form:

$$A_1, \ldots, A_m \Longrightarrow B$$

where the A_i are conjuncts of a formula that are deleted and replaced by B in an application of the rule. So, for example, the simplification rule: $P, \neg P \Longrightarrow \bot$ applied to the conjunct:

$$Q \vee \Diamond(\neg P \wedge Q \wedge P)$$

yields

$$Q \vee \Diamond(Q \wedge \bot).$$

(Note: $\top$ and $\bot$ stand for "true" and "false" respectively.) Deduction rules have the form:

$$A_1, \ldots, A_m \longmapsto B$$

where again the A_i are conjuncts of a formula. In this case the formula B is added as an extra conjunct.

The non-clausal resolution rule is a deduction rule of the form:

$$A\langle C\rangle, B\langle C'\rangle \longmapsto A\theta\langle\top\rangle \vee B\theta\langle\bot\rangle$$

where the notation $A\langle C\rangle$ indicates that C occurs as a subformula of A, and θ is the most general unifier of C and C'. Crucially, to retain soundness, the following restrictions are placed on the application of the resolution rule:

1. *Same world* restriction: the occurrences of C replaced in an application of the resolution rule *must not be in the scope of any modal operators.*

2. The replaced instances of $C\theta$ and $C'\theta$ are *not in the scope of any quantifier* in $A\theta$ or $B\theta$.

(In fact, these are not the only restrictions introduced in [AM86a]. Other, more complicated restrictions are necessary for dealing with quantifier prefixes of A and B. However, is not necessary to repeat the other restrictions for the purposes of the argument below.) Abadi and Manna's system for S5 is shown in Figure 8–2. (The notation $Q^{\forall}$ and $Q^{\exists}$ is used by these authors to indicate quantifier occurrences of universal and existential force within a formula respectively.)

1. Simplification rules:

   - True-false simplification rules:
     - $\top \vee A \Longrightarrow \top$.
     - $\bot, A \Longrightarrow \bot$.
     - $\Diamond \bot \Longrightarrow \bot$.
   - Negation rules:
     - $\neg \Box A \Longrightarrow \Diamond \neg A$.
     - $\neg \Diamond A \Longrightarrow \Box \neg A$.
     - $\neg (A \wedge B) \Longrightarrow (\neg A \vee \neg B)$.
     - $\neg (A \vee B) \Longrightarrow (\neg A \wedge \neg B)$.
     - $\neg \neg A \Longrightarrow A$.
   - Weakening rule:
     - $A, B \Longrightarrow A$.
   - Distribution rule:
     - $A, B_1 \vee \cdots \vee B_m \Longrightarrow (A \wedge B_1) \vee \cdots \vee (A \wedge B_m)$.
   - Quantifier extraction rules:
     - $A\langle Q^{\forall} x.C[x] \rangle \Longrightarrow \forall x'.A\langle C[x'] \rangle$.
     - $A\langle Q^{\exists} x.C[x] \rangle \Longrightarrow \exists x'.A\langle C[x'] \rangle$. (Restriction: the replaced expression should not occur in the scope of any quantifier of universal force $Q^{\forall}$ or modal operator of necessary force.)

2. Deduction rules:

   - The resolution rule (with restrictions).
   - Modal rules:
     - $\Box A, \Diamond B \longmapsto \Diamond (\Box A \wedge B)$.
     - $\Box A \longmapsto A$.
     - $\Diamond A, \Diamond B \longmapsto \Diamond (\Diamond A \wedge B)$.
     - $A \longmapsto \Diamond A$.

Figure 8–2: Abadi and Manna resolution system for S5.

Here is a proof of the formula:

$$\Box(\forall x Px) \Rightarrow (\forall x \Box Px)$$

using this system, (roughly) as presented in [AM86a]. (They actually proved this formula in their system for K.) The first step is to remove the implication and negate the formula in order to derive $\perp$:

$$\neg[\neg\Box(\forall x Px) \vee (\forall x \Box Px)].$$

By the negation rules we get:

$$\Box(\forall x Px) \wedge (\exists x \Diamond \neg Px).$$

By the existential quantifier extraction rule we get:

$$\exists x'[\Box(\forall x Px) \wedge \Diamond \neg Px'].$$

The first modality rule gives us:

$$\exists x'[\Box(\forall x Px) \wedge \Diamond \neg Px' \wedge \Diamond(\Box(\forall x Px) \wedge \neg Px')].$$

Weakening gives us:

$$\exists x' \Diamond[\Box(\forall x Px) \wedge \neg Px')].$$

The second modality rule gives us:

$$\exists x' \Diamond[(\forall x Px) \wedge \neg Px')].$$

Finally, resolution with $A = \neg Px'$, $B = Px$, $C = Px'$ and $C' = Px$ gives us:

$$\exists x' \Diamond[(\forall x Px) \wedge \neg Px') \wedge (\neg\top \vee \perp)].$$

True-false simplification gives us:

Notice the number of rules that need to be applied in order to perform what, in systems based on the matrix characterisations, would be a *single* step performed by unification; *i.e.*, testing the complementarity of a single connection.

Notice that at each stage many of the rules of the system are applicable in multiple places within the proof state (conjunction). But crucially, notice the application of the modal rules to move the atomic formulae into the same modal context so that the resolution rule can apply. The system has all the order dependence problems of the tableau systems, as well as a non-analytic nature that means as the deduction proceeds, the number of possible inferences to choose from at the next stage increases dramatically. Notice the judicious use of the weakening rule to focus attention. Considering the combinatorial problems that would arise in a large search of this kind, we must conclude that the Abadi and Manna systems do not provide bases adequate for automated proof search in modal logics as the matrix characterisations, despite their use of resolution. The redundancies in the systems arise from two sources: the non-clausal resolution rule itself, and the modal rules.

Despite the disadvantages outlined above, their method has an advantage in that the inference rules are locally defined, in contrast to the complex nature of the matrix systems. This supports the rapid extension of their basic modal systems to intensional logics with other types of modal operator, most notably *temporal logics* [AM86b] with many different and interacting forms of modal operator (*eg.*, "next-time" operators *etc*). The extension of the matrix characterisations to such logics requires careful thought and has not been investigated by the author. The Abadi and Manna systems are also somewhat easier to use by hand.

We conclude that the non-clausal modal resolution methods leave much to be desired for automated proof search compared with the matrix systems.

8.4 Hybrid systems.

In the previous section we reviewed two proposals for modal proof systems that attempt to import the efficiency of resolution to modal logics. One was based on a normal form, the other was not. The resolution proposals are separated from

the more standard sequent/tableau approaches by the central rôle played by the connection. The other major component of resolution that makes it suitable for automated proof search is the use of unification to manage the interaction of quantifiers. Modal operators have the same computational properties as quantifiers. The major failing of the resolution proposals was their failure to extend the unification solution to these operators. The modal matrix characterisations were specifically designed with this in mind. In the final analysis, the resolution systems proposed resort to adhoc methods of manipulating modal formulae so as to allow the application of a constrained resolution rule.

In this section we review two slightly more principled approaches to manipulating modalities. While recognising that the classical resolution rule must be restricted in its application for soundness, Wrightson [Wri85] and Konolige [Kon86] fall back on tableau methods to cope with the modalities. The results are an interesting hybrid of tableau and resolution. The problem is that, as Konolige explicitly acknowledges in [Kon86], the manner in which the resolution rule is utilised is not effective. That is to say, arbitrary search may be necessary to determine if a given resolution inference is justified.

8.4.1 Theory resolution and tableaux.

In [Kon86], Konolige develops hybrid systems of resolution and tableaux for the varying domain versions of the modal logics treated in this thesis. We believe that his systems can be extended to the cumulative domain variants, but it is not clear that they extend to the constant domain variants also.

The idea is simple. First, a clausal form is defined in which modal operators are treated as predicate symbols. For example $\Box(\forall x Px \wedge Qy)$ is taken to be a single literal with one free variable. Other than this the details are as in classical logic. A technique of a *bullet* operator is used to allow Skolemisation. For example, the clausal form of the sentence:

$$\forall x \exists y P(x,y) \Rightarrow \Box \exists z Q(x,y,z)$$

is

$$\neg P(x, f(x)) \vee \Box \exists z Q(\bullet x, \bullet f(x), z).$$

The term $\bullet f(x)$ indicates that the term $f(x)$ must be interpreted outside the modal context. It will not unify with a non bulleted term. Similar remarks hold for the Skolem variable $\bullet x$. This is Konolige's method of capturing the varying domain conditions. The matrix characterisations utilise prefixes for this task. We shall restrict our attention to the propositional systems, since this will be sufficient for our central arguments concerning proof search.

Konolige's resolution rule is based on Stickel's theory resolution rule [Sti85a], and is shown for a series of logics in Figure 8–3.

The second rule permits the construction of a resolvent:

$$A_1 \vee A_2 \vee \cdots \vee A_1' \vee A_2' \vee \cdots A_1'' \vee A_2'' \vee \cdots$$

provided the associated set below the rule is unsatisfiable. If the reader thinks of the premises of the second rule as being a subset of the formulae on a tableau (or in a sequent), the rule allows the resolvent to be adjoined to the current tableau provided a new tableau, containing the formulae of the associated set, is unsatisfiable. The system spawns new tableau at each application of theory resolution. A hierarchy of tableaux is thus constructed. The definition of the auxiliary tableau differs from logic to logic. For example, the inclusion of the set: $\Box\Gamma$, reflects the transitivity of the accessibility relation of the logic. It should be compared with the set of formulae "preserved" through the application of a modal sequent rule for the transitive logics described in Chapter 4. In this vein, the first rule is essentially the ν rule of the standard modal sequent calculi. Notice that it is sound only for the reflexive logics.

Each theory resolution step, therefore, is nothing more than an application of modal sequent rules, manipulating the modal context of non-modal formulae. Different modal contexts are represented by different tableau, as in the original tableau systems of Kripke [Kri63]. Ordinary resolution is used to deduce whether a given tableau is unsatisfiable.

This first rule is only for the idealisable logics T, S4 and S5:

$$\frac{\Box B \vee A}{B \vee A}$$

This second rule is for all the modal logics stated:

$$\frac{\begin{array}{c}\Box C_1 \vee A_1 \\ \Box C_2 \vee A_2 \\ \vdots \\ \neg\Box D_1 \vee A'_1 \\ \neg\Box D_2 \vee A'_2 \\ \vdots \\ E_1 \vee A''_1 \\ E_2 \vee A''_2 \\ \vdots\end{array}}{A_1 \vee A_2 \vee \cdots \vee A'_1 \vee A'_2 \vee \cdots A''_1 \vee A''_2 \vee \cdots}$$

where

$$\left.\begin{array}{ll} K,T & \{\,\Gamma, \neg C_1\,\} \\ K4,S4 & \{\,\Gamma, \Box\Gamma, \neg C_1\,\} \\ S5 & \{\,\Gamma, \Box\Gamma, \neg D_1, \neg\Box\Delta, \neg\Box\neg\Sigma\,\} \end{array}\right\} \text{ is unsatisfiable}$$

and

$$\begin{array}{rcl} \Gamma & = & \{\,C_1, C_2, \ldots\,\} \\ \Delta & = & \{\,D_1, D_2, \ldots\,\} \\ \Sigma & = & \{\,E_1, E_2, \ldots\,\} \\ \Box\Gamma & = & \{\,\Box C_1, \Box C_2, \ldots\,\} \\ & etc & \\ & \vdots & \end{array}$$

Figure 8–3: Konolige's theory resolution rule for modal logics.

Notice that the theory resolution rule is not effective. The formation of the resolvent, and hence its subsequent use as a parent of other resolution inferences, is dependent on the unsatisfiability of the auxiliary tableau. This latter question can require arbitrary search. To address this problem, Konolige and Geissler propose interleaving inferences on the auxiliary tableau with inferences on the main tableau. However, notice that there is a choice to be made as to which formulae to include in a given application of theory resolution. The system is prone to all the same redundancies as standard sequent/tableau systems. The only advantage is the use of classical resolution *within* an individual tableau to demonstrate its unsatisfiablility.

We conclude that Konolige's systems are less suitable for automated proof search than the matrix characterisations.

8.4.2 Connections in tableau.

Wrightson [Wri85] recognises the central component of resolution as being the connection, and its major problem the requirement of normal form. He also recognises the utility of tableau/sequent rules for formalising the properties of intensional operators and that tableau suffer from what we have termed: the relevance problem.

He proposes the use of tableau for "non-classical" logics, but augmented with connections, stored in a connection graph [Kow75], to guide the application of tableau rules. His proposals do not address the problems of notational redundancy in tableau, nor the order dependence of the modal tableau rules, but do incorporate unification to overcome the order dependence of the quantifier rules.

8.5 Conclusions.

In this chapter we have reviewed a number of proposals for automated proof search in modal logics. We have argued that the matrix characterisations developed above are more appropriate as a basis for this task. None of the proposals reviewed solve the basic problem arising from the order dependence of modalities, although the resolution based systems do adopt Robinson's unification solution to the order dependence of quantifiers. We are led to the conclusion that there is, in general, a lack of appreciation of the proof-theoretic rôle of unification. We believe that this state of affairs has arisen because of the semantic justification of unification, involving Skolemisation. (Both Abadi and Manna's, and Konolige's proposal introduced a complicated form of modal Skolemisation, though the first two authors leave its use as optional.) If unification had been seen as a proof-theoretic tool for managing the interactions of quantifiers, the solution developed in this part of the thesis for modal logics would have emerged sooner. We point to our elaboration and analysis of the function of unification in Part I of this thesis, based on that given by Bibel in [Bib82a], as a major contribution to the field of automated theorem proving.

Part III

Automated deduction in intuitionistic logic.

Summary.

The main results presented in the thesis so far have been:

Part I: The decomposition of Bibel's Connection Calculus [Bib80,Bib82c]: a matrix characterisation of validity for classical logic, into a set of individual techniques for overcoming problems of redundancy in proof search in that logic.

Part II: The formulation, using these techniques, of matrix characterisations of validity for a wide class of modal logics, and hence the provision of methods for efficient automated proof search in these logics.

The modal logics treated in Part II are *extensions* of classical logic. That is to say, they contain first-order classical logic as a subsystem. In this part of the thesis we reinforce the results obtained so far by presenting a matrix characterisation of validity for first-order intuitionistic logic. Intuitionistic logic is a *subsystem* of classical logic, and is perhaps the archetypal "non-classical" logic. In this way we achieve two objectives: firstly, we provide further evidence of the power of our approach for the efficient automation of proof search in arbitrary logics; secondly, we provide a basis for efficient proof methods for what is possibly the central logic of computation, and of considerable current interest [Mar82,Con86,CH85].

There is only one chapter in this part of the thesis since we keep the details to a minimum. Our method is, as should be apparent by now, to analyse the redundancies within the search space induced by a standard cut-free sequent calculus for intuitionistic logic, and selectively apply the techniques isolated in Part I of this thesis to remove them.

Chapter 9

Matrix proof methods for intuitionistic logic.

9.1 Introduction.

Intuitionistic logic is perhaps the archetypal "non-classical" logic. Hailed by some as the "proper" foundational logic for mathematics (*eg.*, [Bro75,Dum77]), it is currently receiving wide attention within computer science as a means of formalising the notion of "construction" and hence computation (*eg.*, [Mar82]). As with the modal logics of Part II, it is beyond the scope of this thesis to motivate the use of this logic for particular applications, we simply point the reader to the references cited. We note, however, that the logic is not only receiving attention as a tool for the theoretical analysis of computational constructs, it is also proposed as a logic for the *practical* derivation of programs; a formal approach to the construction of verifiably correct software (*eg.*, [Mar82,Con86]). Applications such as in the computer support for these principled methods of program derivation, serve to motivate the need for efficient automated proof search in this logic.

In this chapter we present a matrix characterisation of validity for first-order intuitionistic logic along the lines of the characterisations developed in Part II for modal logics. In fact, our method is the same: we start by presenting the

(Kripke) semantics for intuitionistic logic (§9.2). We then investigate the properties of the search space induced by a standard cut-free sequent calculus for the logic (§9.3), and adapt the matrix techniques accordingly (§9.4). We prove the correctness and completeness of the matrix characterisation by means of an embedding of intuitionistic logic in S4, due to Gödel [God69] (§9.5). Finally we compare proof procedures based on the matrix characterisation with other proposals for automating proof search in this important logic (§9.6).

Throughout, we keep the details to a minimum. The reader will be familiar enough with the development method by now to fill the gaps. In the concluding chapter of this thesis we discuss the import of the material of this chapter for the matrix approach to automating proof search in arbitrary logics. For typographic ease, and following Gentzen [G69], we use the symbol "J" to denote the system of intuitionistic logic.

9.2 Kripke semantics for J.

The language of J is the same as the language of classical logic. The distinction between the two logics arises in the semantics of the connectives and quantifiers. Once again we assume an arbitrary, but fixed set of variables and predicate symbols, but allow the constants to vary, each distinct set defining a new language. We talk of sentences and formulae *over* a set of constants when we wish to emphasise the constants of the language. As in previous chapters we use A, B, C to denote (intuitionistic) formulae.

A first-order intuitionistic frame, or J-frame, is a quadruple: $\langle G, R, D, \bar{D} \rangle$, where

- G is a non-empty set (of points),
- R is a transitive and reflexive relation on G,
- D is a non-empty set (of constants), and

- $\bar{D}$ is a mapping from G to non-empty subsets of D that satisfies the cumulative domain condition: for all $w, w' \in G$,

$$w \; R \; w' \quad \text{implies} \quad \bar{D}(w) \subseteq \bar{D}(w').$$

Without any loss of essential generality we further assume that:

$$D = \bigcup_{w \in G} \bar{D}(w).$$

REMARK. Notice that a J-frame is a cumulative domain S4-frame (and vice versa). (END OF REMARK.)

A first-order intuitionistic (Kripke) model is a quintuple: $\langle G, R, D, \bar{D}, \Vdash \rangle$, where $\langle G, R, D, \bar{D} \rangle$ is a J-frame and $\Vdash$ a relation between elements of G and sentences over D such that:

0. For A atomic, $w \Vdash A$ and $w \; R \; w'$ implies $w' \Vdash A$.

1. $w \Vdash A \wedge B$ iff $w \Vdash A$ and $w \Vdash B$.

2. $w \Vdash A \vee B$ iff either $w \Vdash A$ or $w \Vdash B$.

3. $w \Vdash A \Rightarrow B$ iff for all $w' \in G$, such that $w \; R \; w'$, either $w' \nVdash A$ or $w' \Vdash B$.

4. $w \Vdash \neg A$ iff for all $w' \in G$, such that $w \; R \; w'$, $w' \nVdash A$.

5. $w \Vdash \forall x A$ iff for all $w' \in G$, such that $w \; R \; w'$, and all $c \in \bar{D}(w')$, $w' \Vdash A[c/x]$.

6. $w \Vdash \exists x A$ iff for some $c \in \bar{D}(w)$, $w \Vdash A[c/x]$.

Once again, the relation $w \Vdash A$ can be read "w forces A." Notice that four clauses of the above definition are "modal," in that they refer to the accessibility relation, R, of the frame.

Validity in the language of a model, interpretations, and validity in general are defined as in the modal case. Briefly: A sentence A, of the (intuitionistic)

language over D, is *valid in the model:* $\langle G, R, D, \bar{D}, \Vdash \rangle$, provided, for each $w \in G$ such that the constants of A are in $\bar{D}(w)$, we have $w \Vdash A$. An *interpretation* of a language over D_0 in the model $\langle G, R, D, \bar{D}, \Vdash \rangle$, is a mapping $\iota: D_0 \mapsto D$. A sentence A, of the language over D_0, is *valid under the interpretation* ι in the model just in case $\iota(A)$ is valid in that model. Such a sentence is *valid* just in case it is valid under every interpretation in every model.

9.3 A cut-free sequent calculus for J.

In this section we present a standard cut-free sequent calculus for J, and investigate the search space generated by it. Similar calculi can be found, for instance, in Dummett [Dum77] or Fitting [Fit69]. As usual, we work with sets of formulae, so there are no structural rules. The formulae are over a countable language containing denumerable constant and parameter symbols. The full system is shown in Figure 9–1. This system is both correct and complete for J (see, *eg.*, [Fit69]).

The calculus is similar to the classical sequent system. In fact the two systems differ only in the three rules: $\longrightarrow \neg$, $\longrightarrow \Rightarrow$ and $\longrightarrow \forall$. For these rules the succedent of the premise is restricted to the side formula of the inference, whereas in the corresponding classical rules the succedent may contain multiple formulae. We shall call these rules *special.*

For the purposes of proof search we invert the rules as usual, forming a tableau system. Used in this way, the three special rules cause formulae to be deleted from the sequent; a situation we are familiar with from the modal sequent calculi of Part II. There are differences however. Whereas in the modal calculi the rules that cause such deletions concern the modal operators only, here the special rules are for particular occurrences of connectives and quantifiers. Whereas in the modal sequent calculi the modal operators also provide the means of preserving formulae during an inference involving such "deletion," here there is no specific operator or connective identified for this purpose; all

$$\Gamma, A \longrightarrow A, \Delta$$

$$\frac{\Gamma, A, B \longrightarrow \Delta}{\Gamma, A \wedge B \longrightarrow \Delta} \wedge \longrightarrow \qquad \frac{\Gamma \longrightarrow A, \Delta \quad \Gamma \longrightarrow B, \Delta}{\Gamma \longrightarrow A \wedge B, \Delta} \longrightarrow \wedge$$

$$\frac{\Gamma, A \longrightarrow \Delta \quad \Gamma, B \longrightarrow \Delta}{\Gamma, A \vee B \longrightarrow \Delta} \vee \longrightarrow \qquad \frac{\Gamma \longrightarrow A, B, \Delta}{\Gamma \longrightarrow A \vee B, \Delta} \longrightarrow \vee$$

$$\frac{\Gamma \longrightarrow A, \Delta \quad \Gamma, B \longrightarrow \Delta}{\Gamma, A \Rightarrow B \longrightarrow \Delta} \Rightarrow \longrightarrow \qquad \frac{\Gamma, A \longrightarrow B}{\Gamma \longrightarrow A \Rightarrow B, \Delta} \longrightarrow \Rightarrow$$

$$\frac{\Gamma \longrightarrow A, \Delta}{\Gamma, \neg A \longrightarrow \Delta} \neg \longrightarrow \qquad \frac{\Gamma, A \longrightarrow}{\Gamma \longrightarrow \neg A, \Delta} \longrightarrow \neg$$

$$\frac{\Gamma, A[c/x] \longrightarrow \Delta}{\Gamma, \forall x A \longrightarrow \Delta} \forall \longrightarrow \qquad \frac{\Gamma \longrightarrow A[a/x]}{\Gamma \longrightarrow \forall x A, \Delta} \longrightarrow \forall$$

$$\frac{\Gamma, A[a/x] \longrightarrow \Delta}{\Gamma, \exists x A \longrightarrow \Delta} \exists \longrightarrow \qquad \frac{\Gamma \longrightarrow A[c/x], \Delta}{\Gamma \longrightarrow \exists x A, \Delta} \longrightarrow \exists$$

For the $\longrightarrow \forall$ and $\exists \longrightarrow$ rules, the parameter a must not occur in the conclusion.

Figure 9–1: A cut-free sequent calculus for intuitionistic logic.

antecedent formulae are preserved. Two simple examples will help to illustrate this behaviour. EXAMPLE. The first example is a well-known non-theorem of J: $\neg\neg A \Rightarrow A$. A derivation of this formula is shown below. We consider such derivations to have been constructed from the root upwards. Notice how the leaf of the derivation is not a basic sequent, *i.e.*, the derivation is not a proof. Notice also how the (inverted) $\longrightarrow \neg$ rule causes the "deletion" of all other succedent formulae except the side formula of the inference.

$$\dfrac{\dfrac{\dfrac{A \longrightarrow}{\longrightarrow \neg A, A}\ \longrightarrow \neg}{\neg\neg A \longrightarrow A}\ \neg \longrightarrow}{\longrightarrow \neg\neg A \Rightarrow A}\ \longrightarrow\Rightarrow$$

(END OF EXAMPLE.)

EXAMPLE. The second example is the J-theorem: $A \Rightarrow (B \wedge C) \Rightarrow ((A \Rightarrow B) \vee (A \Rightarrow C))$. A proof of this sentence is shown below.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A \longrightarrow A, B \qquad \dfrac{B, C, A \longrightarrow B}{B \wedge C, A \longrightarrow B}\ \wedge \longrightarrow}{A \Rightarrow (B \wedge C), A \longrightarrow B}\ \Rightarrow \longrightarrow}{A \Rightarrow (B \wedge C) \longrightarrow A \Rightarrow B, A \Rightarrow C}\ \longrightarrow\Rightarrow}{A \Rightarrow (B \wedge C) \longrightarrow (A \Rightarrow B) \vee (A \Rightarrow C)}\ \longrightarrow \vee}{\longrightarrow A \Rightarrow (B \wedge C) \Rightarrow ((A \Rightarrow B) \vee (A \Rightarrow C))}\ \longrightarrow\Rightarrow$$

Here the (inverted) $\longrightarrow\Rightarrow$ rule is applied to a succedent formula: $A \Rightarrow B$, before the antecedent implication: $A \Rightarrow (B \wedge C)$, is reduced with the (inverted) rule $\Rightarrow\longrightarrow$. The effect of applying the latter rule first is shown below.

$$\longrightarrow\Rightarrow\ \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A \longrightarrow B}{\longrightarrow A, A \Rightarrow B, A \Rightarrow C} \qquad \dfrac{\dfrac{B, C, A \longrightarrow B}{B \wedge C, A \longrightarrow B}\ \wedge \longrightarrow}{B \wedge C \longrightarrow A \Rightarrow B, A \Rightarrow C}\ \longrightarrow\Rightarrow}{A \Rightarrow (B \wedge C) \longrightarrow A \Rightarrow B, A \Rightarrow C}\ \Rightarrow\longrightarrow}{A \Rightarrow (B \wedge C) \longrightarrow (A \Rightarrow B) \vee (A \Rightarrow C)}\ \longrightarrow \vee}{\longrightarrow A \Rightarrow (B \wedge C) \Rightarrow ((A \Rightarrow B) \vee (A \Rightarrow C))}\ \longrightarrow\Rightarrow$$

Notice that although we successfully construct a basic sequent in the right-hand branch of the derivation as before, the left-hand branch presents a problem. The reader should recognise this as a classic case of order dependence amongst the

rules. Notice also that this dependence arises in the propositional fragment of the calculus.

The reader should also note the repetition of formula within the sequent derivations and the emphasis on connectives. (END OF EXAMPLE.)

The two examples above serve to illustrate that the cut-free sequent calculus presented in Figure 9–1 suffers from the same three classes of redundancy as the other calculi we have studied in this thesis, namely:

- notational redundancy,
- relevance, and
- order dependence amongst the rules.

The first redundancy arises (as usual) from the basic sequent or tableau framework. The second type of redundancy arises (as usual) from the emphasis on connectives. The third, and most interesting redundancy, arises from the semantics of the connectives themselves.

The special rules for certain occurrences of the connectives: $\neg$ and $\Rightarrow$, and the universal quantifier $\forall$, induce one sort of order dependence. A second source of order dependence is the interaction between quantifiers of universal and existential force arising from the parameter condition on the rules: $\longrightarrow \forall$ and $\exists \longrightarrow$. This latter problem is common to all the quantified logics we have studied in this thesis. The two sorts of dependencies will, of course, interact. Consequently the situation resembles that of quantified modal logics.

9.4 A matrix characterisation of validity in J.

In this section we develop a matrix characterisation of validity for intuitionistic logic. We shall keep the details to a minimum as the reader will have seen enough of this already to fill in the gaps. We present motivational arguments in the following subsection and in short remarks thereafter. This section is not intended to be self contained. We rely heavily on definitions given in Parts I and II. In the next section we prove the correctness and completeness of the characterisation.

9.4.1 Overview.

First notice that the sequent calculus for J presented above possesses the *sub-formula property*. Moreover, the basic sequent is common to the classical and modal calculi. From these observations we conclude that our standard approach for removing the notational and relevance redundancies of sequent calculi are applicable in the case of J. We expect, therefore, to formulate the characterisation in terms of the positions of formula trees that capture the structure of formulae and support the use of structure sharing techniques in practical implementations. We also expect the notion of *path* and *connection* to be unchanged from the classical and modal cases. The validity of a formula will be characterised in terms of a *spanning* set of *complementary* connections within it.

The most interesting problem, as we found with modal logics in Part II, is to formulate an appropriate notion of complementarity that takes account of the order dependence amongst rules. We have noted that the "special" rules induce an order dependence in the sequent search space in addition to the expected order dependence induced by the quantifier rules. We know how to deal with the quantifier problem from our experience with classical logic. The former problem seems to present us with a new challenge however. Superficially it resembles the order dependence induced by the modal operators in that S-formulae are deleted

by the application of certain (inverted) rules. In contrast to the modal case, there is not one operator and its dual responsible for this problem. Indeed, it is certain occurrences of connectives and quantifiers that contribute to the problem. In addition, there appears to be no identified operator by which formulae are "preserved" through an application of the special rules; in fact, all antecedent formulae are preserved through such an application.

This latter observation is the clue. We have already developed the notion of *polarity* which allows us to identify antecedent subformulae in terms of the structure of the formula whose validity we are testing. Moreover, the only connectives that move S-formulae from antecedent to succedent, and vice versa, are the rules for implication and negation. We must ensure that formula occurrences containing the two identified atomic formulae of a connection as subformulae are not deleted by the application of a special rule. This can be achieved by ensuring that the "ancestors" of the atomic formulae (those formulae of which the atomic formulae are subformulae) are antecedent formulae during such applications. To do this, we only have need to control the application of the rules dual to the special rules, namely: $\neg \longrightarrow$, $\Rightarrow \longrightarrow$ and $\forall \longrightarrow$.

The final issue concerns the genericity of quantifiers and antecedent formulae. Once again we employ a *multiplicity* to encode the number of "copies" of subformulae utilised in a derivation. To support a clean technical solution to the duplication of antecedent formulae we alter the definition of a formula tree slightly by adding extra positions. It turns out that this extension mirrors the fundamental relationship between J and S4. First we review some uniform notation.

9.4.2 Uniform notation.

Signed formulae are defined as usual. We use Smullyan and Fitting's classification of signed formulae developed for classical logic to capture the structural properties of the connectives and quantifiers. The classification is repeated in

α	α_1	α_2
$\langle A \wedge B, 1\rangle$	$\langle A, 1\rangle$	$\langle B, 1\rangle$
$\langle A \vee B, 0\rangle$	$\langle A, 0\rangle$	$\langle B, 0\rangle$
$\langle A \Rightarrow B, 0\rangle$	$\langle A, 1\rangle$	$\langle B, 0\rangle$
$\langle \neg A, 1\rangle$	$\langle A, 0\rangle$	$\langle A, 0\rangle$
$\langle \neg A, 0\rangle$	$\langle A, 1\rangle$	$\langle A, 1\rangle$

γ	$\gamma_0(a)$
$\langle \forall x A, 1\rangle$	$\langle A[a/x], 1\rangle$
$\langle \exists x A, 0\rangle$	$\langle A[a/x], 0\rangle$

β	β_1	β_2
$\langle A \wedge B, 0\rangle$	$\langle A, 0\rangle$	$\langle B, 0\rangle$
$\langle A \vee B, 1\rangle$	$\langle A, 1\rangle$	$\langle B, 1\rangle$
$\langle A \Rightarrow B, 1\rangle$	$\langle A, 0\rangle$	$\langle B, 1\rangle$

δ	$\delta_0(a)$
$\langle \forall x A, 0\rangle$	$\langle A[a/x], 0\rangle$
$\langle \exists x A, 1\rangle$	$\langle A[a/x], 1\rangle$

Table 9–1: Uniform notation for signed intuitionistic formulae.

Table 9–1 for the reader's convenience. As before, we use X, Y, Z to denote arbitrary signed formulae.

In addition, we call a (signed) formula *special* if it is atomic, or its major connective/quantifier is either an implication, a negation, or a universal quantifier.

9.4.3 Formula occurrences.

The notion of *formula tree*, *positions*, *labels* and *polarity* are defined roughly as before. There is one crucial difference. Let A be an intuitionistic formula and X the signed formula $\langle A, 0\rangle$. In the classical and modal cases the formula tree for X contained one position for each distinct subformula of A. Here we add extra positions for technical convenience. Basically, an extra position is inserted between each position that corresponds to a special subformula of A and its original parent. The polarity and label of such an extra position is defined to be the same as the polarity and label of its child. We erect a classification for

these extra positions as follows. If the polarity of such a position is 1, it has principal type ϕ; if its polarity is 0, its principal type is ψ. The secondary types of these extra positions is obtained from their parents as usual. The child of such a position is given secondary type ϕ_0 or ψ_0 depending on the principal type of its parent. In addition, we classify the root position as having secondary type ψ_0 (cf. the modal case where the root position was defined to have π_0 secondary type). *Atomic* positions are positions labelled by atomic formulae, but *not* of ϕ or ψ type. (We need this extra caveat because we defined the label of the extra positions, *i.e.*, positions of ϕ and ψ type, to be the label of their child. Consequently there are non-leaf positions of the formula tree labelled by atomic formulae. The caveat ensures that only leaf positions are deemed to be atomic.)

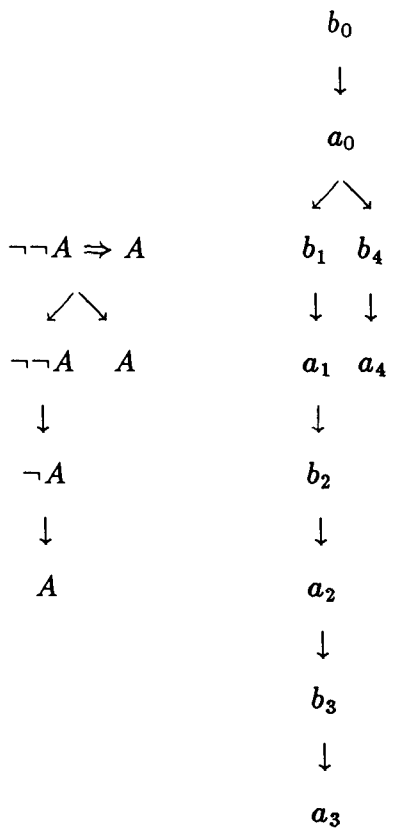
An example will help. A formula tree for the signed formula: $\langle \neg\neg A \Rightarrow A, 0 \rangle$, is shown in Figure 9–2. The table in the figure contains the appropriate classification of positions.

Once again we shall use the capital Greek letter to denote the set of positions of that type in a given formula tree. For the example tree of Figure 9–2, $\Phi_0 = \{a_1, a_3\}$ and $\Psi = \{b_0, b_2, b_4\}$. We say that the positions of secondary type ϕ_0 and ψ_0 are *special positions.*

REMARK. Notice that by the addition of the extra positions we have ensured that all special positions have secondary type ϕ_0 or ψ_0 depending on their polarity. This is the motivation for the addition of these positions. In the next section we define a multiplicity to capture the genericity of antecedent formulae using this classification. In following sections, we use this property to define the notion of a prefix. Finally, when we come to prove the correctness and completeness of the characterisation, we shall see how the addition of this structure has semantic significance. (END OF REMARK.)

9.4.4 Multiplicities.

Since the logic is quantified we use a *first-order* multiplicity, μ_Q, to encode the number of distinct copies of subformulae quantified by a quantifier of universal



k	pol(k)	lab(k)	Ptype(k)	Stype(k)
b_0	0	$\neg\neg A \Rightarrow A$	ψ	ψ_0
a_0	0	$\neg\neg A \Rightarrow A$	α	ψ_0
b_1	1	$\neg\neg A$	ϕ	α_1
a_1	1	$\neg\neg A$	α	ϕ_0
b_2	0	$\neg A$	ψ	α_1
a_2	0	$\neg A$	α	ψ_0
b_3	1	A	ϕ	α_1
a_3	1	A	–	ϕ_0
b_4	0	A	ψ	α_2
a_4	0	A	–	ψ_0

Figure 9–2: Formula tree for signed formula: $\langle \neg\neg A \Rightarrow A, 0 \rangle$.

force (γ type) are being utilised in the derivation. Such a multiplicity is a function from Γ_0 to the natural numbers as before.

We define the notion of an *intuitionistic multiplicity* as a function from Φ_0 to the natural numbers. Notice that the sets Γ_0 and Φ_0 are disjoint by virtue of the extra positions. An intuitionistic multiplicity, μ_J, encodes the number of copies of antecedent formulae utilised in a derivation. As in the modal case, a *multiplicity* for a signed formula X is the union of both a first-order and intuitionistic multiplicity. We shall again use μ, possibly subscripted, to denote (combined) multiplicities.

The *indexed formula tree* for the indexed formula X^μ is formed from indexed positions as in the classical and modal cases. The notions of labels, polarity and tree ordering, are extended to indexed positions in the usual way. (Recall that the labels of indexed positions contain positions in place of quantified variables.) Indexed positions, k^κ, inherit the types of their underlying (unindexed) components as before. Once again we shall use u, v, possibly subscripted to denote indexed positions when we are not interested in their indices.

IMPORTANT NOTATIONAL POINT. Once more we warn the reader that we shall systematically abuse our notation and use the names of types to denote arbitrary (un)indexed positions of that type within formal definitions such as the definition of the notion of path in the next subsection. In particular, if we say: "if s, ϕ^κ is a path..." we mean that "if s, u is a path, and $\mathrm{Ptype}(u) = \phi$,..." Furthermore, in this context we shall use ϕ_0 to denote the child of u. Similar abuses are extended to the other types. We shall include indices explicitly where necessary. (END OF POINT.)

9.4.5 Paths and connections.

The definition of paths through an indexed formula is an extension of the definition for classical logic to take account of the two new types of position. Let X^μ be an indexed signed formula. A *path* through X^μ is a subset of the positions of its formula tree defined below. We shall again use s and t, possibly subscripted,

to denote paths, and once more adopt the convention that s, u denotes the path $s \cup \{u\}$. The set of paths through X^μ is the smallest set such that:

1. $\{ k_0 \}$ is a path, where k_0 is the root position of the formula tree for X^μ;

2. if s, α^κ is a path, so is $(s \setminus \{\alpha^\kappa\}), \alpha_1{}^\kappa, \alpha_2{}^\kappa$;

3. if s, β^κ is a path, so are $(s \setminus \{\beta^\kappa\}), \beta_1{}^\kappa$ and $(s \setminus \{\beta^\kappa\}), \beta_2{}^\kappa$;

4. if s, γ^κ is a path, so is $s, \gamma_0{}^{\kappa j}$, for any j, $1 \leq j \leq \mu_Q(\gamma_0)$;

5. if s, δ^κ is a path, so is $(s \setminus \{\delta^\kappa\}), \delta_0{}^\kappa$.

6. if s, ϕ^κ is a path, so is $s, \phi_0{}^{\kappa j}$, for any j, $1 \leq j \leq \mu_J(\phi_0)$;

7. if s, ψ^κ is a path, so is $(s \setminus \{\psi\}), \psi_0{}^\kappa$.

REMARK. Notice that the ϕ positions act in a similar way to the generative γ positions.

A more striking similarity can be seen between the clauses for ϕ and ψ and the clauses given for ν and π in Chapter 6. Reading ν for ϕ, π for ψ and σ_M for σ_J, renders the two definitions of path identical. This is not a coincidence, as we shall see in §9.5. (END OF REMARK.)

For a path s through X^μ, the notions

$\mathcal{S}(s)$: the set of positions associated with the path s; and its dual,

$\mathcal{D}(s)$: those positions "reachable" by further path reductions from s,

are defined as for classical logic. *Atomic paths* are also defined as in classical logic and idealisable modal logics, namely those paths, s, for which $\mathcal{D}(s) = \emptyset$.

A *connection* in an (indexed) formula, as before, is a subpath of a path through the formula consisting of two atomic positions of different polarities, but labelled by an atomic formula with the same predicate symbol.

9.4.6 Complementarity.

We now come to the crucial issue, namely, characterising when it is that a connection can be deemed complementary in J, and hence correspond to an instance of the basic sequent. We follow the pattern established for modal logics.

Let A be an intuitionistic formula, X the signed modal formula $\langle A, 0\rangle$ and μ a multiplicity for X. The following definitions are given for a particular indexed formula tree for X^{μ}.

Let $T_J(\mu)$ denote the union of Φ_0 and Ψ_0. (cf. $T_Q(\mu)$ and $T_M(\mu)$.) We associate with each position u of the formula tree a sequence of positions, $\mathrm{pre}(u)$, called a *prefix*, in exactly the same way as was done for modal logic in Part II. That is, if $u_1 \ll u_2 \ll \cdots \ll u_n \lessdot u$, $1 \leq n$, are those elements of $T_J(\mu)$ that dominate u in the formula tree, then

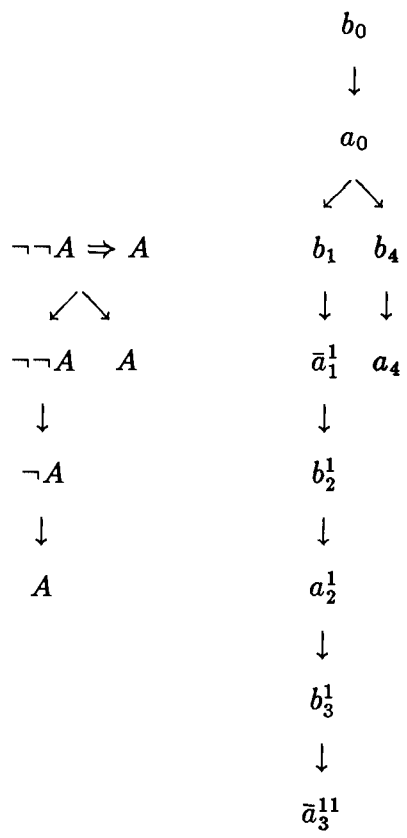
$$\mathrm{pre}(u) = u_1 u_2 \cdots u_n.$$

The prefix of a position encodes the context of that position within the formula tree with respect to the special positions.

Let us update the example. An indexed formula tree for the signed formula: $\langle \neg\neg A \Rightarrow A, 0\rangle$, based on the unindexed formula tree of Figure 9–2 is shown in Figure 9–3. The multiplicity is constant for all γ_0 and ϕ_0 type positions, and equal to 1. The prefixes of the positions are also shown in the table. Notice that we have distinguished the ϕ_0 type positions from the others by means of an overbar.

The final step is to define complementarity for connections in terms of the prefixes of the atomic positions that comprise them. In fact the definition is *exactly the same* as that given for cumulative domain S4 in Part II, with $T_J(\mu)$ playing the rôle of $T_M(\mu)$. We repeat the definition for the reader's convenience.

Accessibility on prefixes. The J-accessibility relation, R_0, on $T_J(\mu)^*$ is defined to be the smallest relation such that: for $p, q \in T_J(\mu)^*$, $p\ R_0\ q$ just in case: either



u	pol(u)	lab(u)	pre(u)	Ptype(u)	Stype(u)
b_0	0	$\neg\neg A \Rightarrow A$	b_0	ψ	ψ_0
a_0	0	$\neg\neg A \Rightarrow A$	b_0a_0	α	ψ_0
b_1	1	$\neg\neg A$	b_0a_0	ϕ	α_1
$\bar{a}_1^1$	1	$\neg\neg A$	$b_0a_0\bar{a}_1^1$	α	ϕ_0
b_2^1	0	$\neg A$	$b_0a_0\bar{a}_1^1$	ψ	α_1
a_2^1	0	$\neg A$	$b_0a_0\bar{a}_1^1a_2^1$	α	ψ_0
b_3^1	1	A	$b_0a_0\bar{a}_1^1a_2^1$	ϕ	α_1
$\bar{a}_3^{11}$	1	A	$b_0a_0\bar{a}_1^1a_2^1\bar{a}_3^{11}$	–	ϕ_0
b_4	0	A	b_0a_0	ψ	α_2
a_4	0	A	$b_0a_0a_4$	–	ψ_0

Figure 9–3: Indexed formula tree for $\langle \neg\neg A \Rightarrow A, 0 \rangle$.

(a) $q = pu$, where $u \in T_J(\mu)$, or

(b) $q = p$, or

(c) $p \prec q$.

As with S4, this can be shortened to:

$$p \, R_0 \, q \quad \text{iff} \quad p \preceq q.$$

Intuitionistic substitutions. An *intuitionistic substitution* is a mapping σ_J from $\Phi_0(\mu)$ to $T_J(\mu)^*$. It induces a relation $\sqsubset_J$ and $\sim_J$ on $T_J(\mu) \times T_J(\mu)$ in the usual way (see Chapter 6, §6.2.4.1).

First-order substitutions. A *first-order substitution* is a mapping σ_Q, from $\Gamma_0(\mu)$ to $T_Q(\mu) \cup C$, where C is the set of constants in the formula being tested for validity. It induces a relation $\sqsubset_Q$ and $\sim_Q$ on $T_Q(\mu) \times T_Q(\mu)$ in the usual way.

J-Admissible substitutions. A *combined substitution* is a pair, $\langle \sigma_J, \sigma_Q \rangle$, consisting of an intuitionistic substitution and a first-order substitution. It is *J-admissible* provided:

1. σ_J respects the J-accessibility relation.

2. $\lhd = (\ll \cup \sqsubset_J \cup \sqsubset_Q)^+$ is irreflexive.

3. If $\sigma_Q(u) = v$, then either:

   (a) $\sigma_J^{\#}(\text{pre}(v)) = \sigma_J^{\#}(\text{pre}(u))$; or

   (b) $\sigma_J^{\#}(\text{pre}(v)) \; R_0 \; \sigma_J^{\#}(\text{pre}(u))$.

   (This latter condition can be simplified to:

   $$\sigma_J^{\#}(\text{pre}(v)) \preceq \sigma_J^{\#}(\text{pre}(u)).)$$

In the above conditions, $\sigma_J^{\#}$ indicates the homomorphic extension of σ_J to $T_J(\mu)^*$.

Complementarity. A connection $\{ u, v \}$ is said to be σ-complementary just in case:

1. $\sigma_J^{\#}(\text{pre}(u)) = \sigma_J^{\#}(\text{pre}(v))$, and

2. $\sigma_Q(\text{lab}(u)) = \sigma_Q(\text{lab}(v))$.

The reader can check that the above definitions are identical to those given in Chapter 6 for cumulative domain S4. In the next section we prove the theorem:

THEOREM 9.1 *An intuitionistic formula A is J-valid if and only if there is a multiplicity, μ, for the signed formula $\langle A, 0\rangle$, a J-admissible combined substitution, σ, and a set of σ-complementary connections that span $\langle A, 0\rangle^{\mu}$.*

We shall end this section with two examples.

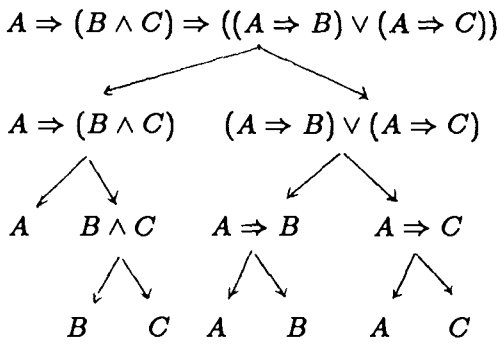
EXAMPLE. Consider the indexed formula tree for $\langle \neg\neg A \Rightarrow A, 0\rangle$ shown in Figure 9–3. There is only one atomic path through the formula and it contains both atomic positions $\bar{a}_3^{11}$ and a_4. These positions form a connection. The prefix of the former is $b_0 a_0 \bar{a}_1^1 a_2^1 \bar{a}_3^{11}$, the prefix of the latter is $b_0 a_0 a_4$. We can compute J-admissible substitutions using the same unification method as was suggested for use in S4. The prefixes cannot be unified since $b_0 a_0 a_4$ is "ground" and does not contain the "ground" position a_2^1. The connection is not J-complementary. From the non-complementary atomic path we can construct a model in which the formula fails to be forced. We leave the details to the reader. (END OF EXAMPLE.)

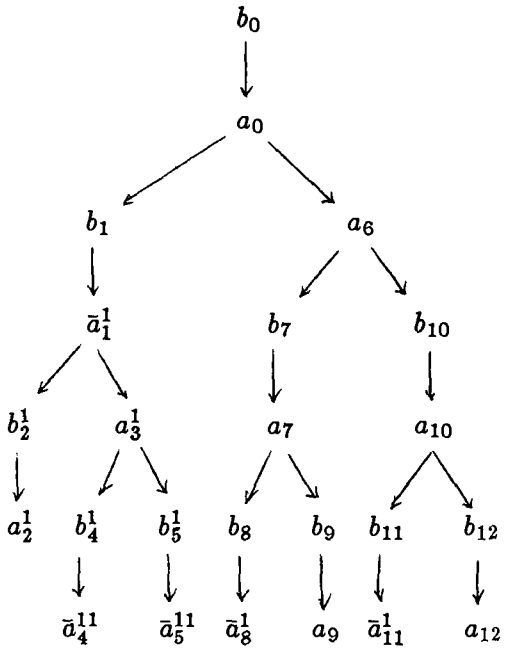
EXAMPLE. In the second example we prove the J-theorem: $A \Rightarrow (B \wedge C) \Rightarrow ((A \Rightarrow B) \vee (A \Rightarrow C))$. An indexed formula tree for the signed formula:

$$\langle A \Rightarrow (B \wedge C) \Rightarrow ((A \Rightarrow B) \vee (A \Rightarrow C)), 0\rangle$$

is shown in Figure 9–4. We have taken a constant multiplicity of 1. Notice that we have only included the significant positions in the table.

There are two atomic paths through this formula, the atomic elements of which can be seen if the formula is displayed in matrix form; *i.e.*, the components





u	pol(u)	lab(u)	pre(u)	Ptype(u)	Stype(u)
b_0	0	$A \Rightarrow (B \wedge C) \Rightarrow ((A \Rightarrow B) \vee (A \Rightarrow C))$	b_0	ψ	ψ_0
a_0	0	$A \Rightarrow (B \wedge C) \Rightarrow ((A \Rightarrow B) \vee (A \Rightarrow C))$	$b_0 a_0$	α	ψ_0
$\bar{a}_1^1$	1	$A \Rightarrow (B \wedge C)$	$b_0 a_0 \bar{a}_1^1$	β	ϕ_0
a_2^1	0	A	$b_0 a_0 \bar{a}_1^1 a_2^1$	–	ψ_0
a_3^1	1	$B \wedge C$	$b_0 a_0 \bar{a}_1^1$	α	β_2
$\bar{a}_4^{11}$	1	B	$b_0 a_0 \bar{a}_1^1 \bar{a}_4^{11}$	–	ϕ_0
$\bar{a}_5^{11}$	1	C	$b_0 a_0 \bar{a}_1^1 \bar{a}_5^{11}$	–	ϕ_0
a_6	0	$(A \Rightarrow B) \vee (A \Rightarrow C)$	$b_0 a_0$	α	α_2
a_7	0	$A \Rightarrow B$	$b_0 a_0 a_7$	α	ψ_0
$\bar{a}_8^1$	1	A	$b_0 a_0 a_7 \bar{a}_8^1$	–	ϕ_0
a_9	0	B	$b_0 a_0 a_7 a_9$	–	ψ_0
a_{10}	0	$A \Rightarrow C$	$b_0 a_0 a_{10}$	α	ψ_0
$\bar{a}_{11}^1$	1	A	$b_0 a_0 a_{10} \bar{a}_{11}^1$	–	ϕ_0
a_{12}	0	C	$b_0 a_0 a_{10} a_{12}$	–	ψ_0

$$\left(\left(\begin{array}{c} A \\ \Rightarrow \\ (B \wedge C) \end{array} \right) \Rightarrow \begin{array}{cc} ((A \Rightarrow B) & (A \Rightarrow C)) \end{array} \right)$$

Figure 9–4: Indexed formula tree for second example.

of α type subformulae are placed horizontally on the page, and the components of β type subformulae are placed vertically on the page. The resulting matrix is also shown in Figure 9–4.

Choose an element from the first column, say a_2^1, representing the occurrence of A in the implication $A \Rightarrow (B \wedge C)$. Search for an atomic position with opposite polarity, but the same predicate symbol in the atomic paths that contain a_2^1. There is a choice: either $\bar{a}_8^1$ or $\bar{a}_{11}^1$. Since these two positions are elements of both atomic paths it doesn't matter which we choose (in this instance). Choose the first: $\bar{a}_8^1$. The prefix of a_2^1 is $b_0 a_0 \bar{a}_1^1 a_2^1$, and the prefix of $\bar{a}_8^1$ is $b_0 a_0 a_7 \bar{a}_8^1$. They are identified by the substitution: $\sigma_J(\bar{a}_1^1) = a_7$ and $\sigma_J(\bar{a}_8^1) = a_2^1$. Provided this substitution is J-admissible we have only to find a connection within the remaining atomic path.

Choose an element from the first column contained in the other path. We have a choice, either $\bar{a}_4^{11}$ or $\bar{a}_5^{11}$. We choose the first possibility (and come back to consider the second later). Having chosen $\bar{a}_4^{11}$ there is no choice for a potential complementary position, the only candidate is a_9. The prefix of $\bar{a}_4^{11}$ is $b_0 a_0 \bar{a}_1^1 \bar{a}_4^{11}$, but under the current intuitionistic substitution becomes $b_0 a_0 a_7 \bar{a}_4^{11}$. The prefix of a_9 is $b_0 a_0 a_7 a_9$. The two can be unified by extending the current substitution with $\sigma_J(\bar{a}_4^{11}) = a_9$.

Both paths contain one of the two connections: $\{a_2^1, \bar{a}_8^1\}$ and $\{\bar{a}_4^{11}, a_9\}$. The reduction ordering induced by the substitution is shown in Figure 9–5. Notice that it is acyclic. The other conditions are easy to check and we leave the details to the reader. Assuming the correctness of the characterisation, the formula is valid. These two connections correspond to the basic sequents in the sequent proof of this formula given in §9.3. Notice how the reduction ordering captures the constraint we discussed there: that the succedent formula $A \Rightarrow B$ be reduced before the antecedent formula $A \Rightarrow (B \wedge C)$. In terms of positions, a_7 must be reduced before $\bar{a}_1^1$.

Recall that at the second stage there was a choice between $\bar{a}_4^{11}$ and $\bar{a}_5^{11}$. We chose the former. Suppose we had chosen the latter. The prefix of $\bar{a}_5^{11}$ is $b_0 a_0 \bar{a}_1^1 \bar{a}_5^{11}$, which under the current substitution becomes $b_0 a_0 a_7 \bar{a}_5^{11}$. The only

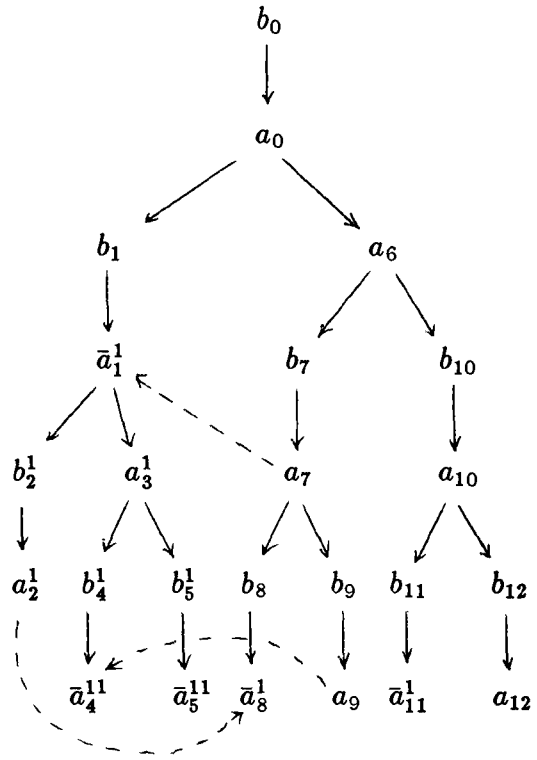


Figure 9–5: Reduction ordering for connections.

possible choice for the complement is a_{12}. The prefix of this position is $b_0a_0a_{10}a_{12}$. The two prefixes cannot therefore be unified. This represents the fact that whichever of the two succedent implications we reduce, the special rule $\longrightarrow\Rightarrow$ causes the deletion of the other. The atomic formulae in the basic sequents of any proof of this formula must involve atoms from one or other of the implications, but not both. (END OF EXAMPLE.)

9.5 Correctness and completeness.

In this section we prove the correctness and completeness of the characterisation of J-validity developed above. We do not prove Theorem 9.1 from first principles. Instead we utilise an embedding of J in cumulative domain S4.

One way of seeing the relationship with modal ideas is to reformulate the semantic clauses for the intuitionistic forcing relation given in the definition of a J-model in §9.2. Notice that the "modal" clauses are those for atomic formulae, negation, implication and universal quantification:

0. For A atomic, $w \Vdash A$ and $w\ R\ w'$ implies $w' \Vdash A$.

3. $w \Vdash A \Rightarrow B$ iff for all $w' \in G$, such that $w\ R\ w'$, either $w' \nVdash A$ or $w' \Vdash B$.

4. $w \Vdash \neg A$ iff for all $w' \in G$, such that $w\ R\ w'$, $w' \nVdash A$.

5. $w \Vdash \forall x A$ iff for all $w' \in G$, such that $w\ R\ w'$, and all $c \in \bar{D}(w')$, $w' \Vdash A[c/x]$.

Compare these clauses with the semantic clause for the modal operator $\Box$ from Part II:

$w \Vdash \Box A$ iff for all $w' \in G$, such that $w\ R\ w'$, $w' \Vdash A$.

If we apply this clause to the formula $\Box\neg A$ *within a modal logic* we obtain:

$w \Vdash \Box\neg A$ iff for all $w' \in G$, such that $w\ R\ w'$, $w' \Vdash \neg A$.

Applying the *classical/modal* clause for negation, the above is equivalent to:

$w \Vdash \Box\neg A$ iff for all $w' \in G$, such that $w\ R\ w'$, $w' \nVdash A$,

which is reminiscent of the intuitionistic clause for negation. A similar analysis can be carried out for the other "modal" clauses of the intuitionistic definition. We leave the details to the reader. In all cases the intuitionistic connective/quantifier resembles its classical counterpart provided we preface it with the modal operator $\Box$. Given, the fact that a J-frame is identical to a cumulative domain S4-frame we might expect that there is a fundamental relationship between cumulative domain S4 and J.

The relationship was identified by Gödel in [God69]. He showed that the set of intuitionistic formulae could be embedded in the set of modal formulae by a mapping $\mathcal{F}$ such that:

THEOREM 9.2 (GÖDEL) *A formula of intuitionistic logic, A, is J-valid if and only if $\mathcal{F}(A)$ is S4-valid.*

We shall use a modification of the embedding taken from Fitting [Fit83]. It is exactly as outlined above with reference to the semantic clauses:

0. $\mathcal{F}(A) = \Box A$, for atomic A.

1. $\mathcal{F}(A \wedge B) = \mathcal{F}(A) \wedge \mathcal{F}(B)$.

2. $\mathcal{F}(A \vee B) = \mathcal{F}(A) \vee \mathcal{F}(B)$.

3. $\mathcal{F}(A \Rightarrow B) = \Box(\mathcal{F}(A) \Rightarrow \mathcal{F}(B))$.

4. $\mathcal{F}(\neg A) = \Box\mathcal{F}(A)$.

5. $\mathcal{F}(\forall x A) = \Box\forall x\mathcal{F}(A)$.

6. $\mathcal{F}(\exists x A) = \exists x\mathcal{F}(A)$.

Let A be an intuitionistic formula and B be a modal formula. Write $\vdash_J A$ for the relation: there is a multiplicity, μ, for $\langle A, 0\rangle$, a J-admissible combined substitution, σ, and a set, U, of σ-complementary connections that span $\langle A, 0\rangle^\mu$. Similarly, write $\vdash_{S4} B$ for the relation: there is a multiplicity, μ', for $\langle B, 0\rangle$, a S4-admissible combined substitution, σ', and a set, U', of σ-complementary connections that span $\langle B, 0\rangle^{\mu'}$. Write $\models_J A$ for A is J-valid and $\models_{S4} B$ for B is S4-valid.

Given the theorem:

THEOREM 9.3 *$\vdash_J A$ if and only if $\vdash_{S4} \mathcal{F}(A)$.*

we are done since:

$$\begin{aligned} \vdash_J A \quad &\text{iff} \quad \vdash_{S4} \mathcal{F}(A) \\ &\text{iff} \quad \models_{S4} \mathcal{F}(A) \\ &\text{iff} \quad \models_J A. \end{aligned}$$

The first step is by Theorem 9.3, the second by the correctness and completeness of the matrix characterisation of validity for S4 proved in Part II, and the final step by Theorem 9.2.

We do not so much as prove Theorem 9.3, as invite the reader to notice that under the definitions given in §9.4, the intuitionistic formula tree for an intuitionistic formula A is isomorphic to the modal formula tree for the modal formula $\mathcal{F}(A)$. The isomorphism identifies ϕ and ψ positions with ν and π positions respectively. Under this isomorphism we have:

- The set of J-paths through $\langle A, 0\rangle^\mu$ are exactly the set of S4-paths through $\langle \mathcal{F}(A), 0\rangle^\mu$.

- The J-atomic positions of $\langle A, 0\rangle^\mu$ are exactly the S4-atomic positions of $\langle \mathcal{F}(A), 0\rangle^\mu$.

- The J-prefix of a position of $\langle A, 0\rangle^\mu$ is exactly the S4-prefix of the corresponding position of $\langle \mathcal{F}(A), 0\rangle^\mu$ under the isomorphism.

- A J-admissible mapping is S4-admissible, and vice versa.

For example,

$$\mathcal{F}(\neg\neg A \Rightarrow A) = \Box(\Box\neg\Box\neg\Box A \Rightarrow \Box A).$$

The embedding $\mathcal{F}$ introduces an "extra" position as the parent of each position corresponding to a special subformula of the intuitionistic formula, exactly as we did in §9.4.

9.6 Related work.

There have been few attempts to automate proof search in intuitionistic logic. There are of course standard methods such as tableau [Fit83] and sequent calculi [G69], but we have already shown that connective-based search methods are quite redundant. Fariñas-del-Cerro [Far86] cites a forthcoming thesis in which his method for modal logics is used to automate proof search in propositional intuitionistic logic via a modal translation (like ours). We reviewed his clausal resolution technique in Chapter 8 and concluded that it was quite redundant and doesn't extend to the full modal language. Presumably this is why his method only extends to the propositional fragment of intuitionistic logic.

Intuitionistic logic is being used within the Logic Programming community, *eg.*, [Gab85,McC86]. The proof methods used are tailored to give a programming language flavour. None of these authors consider the full first-order language and the proof procedures of Gabbay and McCarty are based on tableaux.

The author is currently unaware of any proposal in the literature for a proof procedure for full first-order intuitionistic logic which approaches the efficiency of the one outlined in this chapter. We stress: the standard path-checking algorithms for classical logic are rendered applicable to intuitionistic logic *without change* by the matrix method developed here. A small cost is paid in the increased complexity of the complementarity test.

9.7 Summary.

In this chapter we have defined a matrix characterisation of validity for first-order intuitionistic logic, and proved it correct and complete. We developed the characterisation in the (by now) standard fashion of analysing the properties of a cut-free sequent calculus for the logic. We have shown that the efficiency of proof procedures based on this characterisation compares favourably with other proof methods suggested in the literature for automating proof search in this logic.

The particular characterisation presented is probably not optimal. The formulation was partially led by the desire for a simple correctness and completeness proof via an embedding of J in S4. The particular embedding used was taken from Fitting [Fit83]. There are, however, other embeddings which may lead to less redundant characterisations by introducing fewer "extra" positions in certain circumstances. Such issues have not been explored by the author to-date.

It can be argued that the characterisation developed above is equivalent to first embedding an intuitionistic formula into the modal language and then using the matrix characterisation for cumulative domain S4 on the result. This is true. We have preferred to perform the embedding in the metatheory, rather than the object language, because, as with modal logic, the metatheoretic approach gives us the opportunity to refine the characterisation by "building-in" special techniques based on properties of the logic. We mentioned above that there are a number of embeddings of J in S4. We would hope to be able to take advantage of the best features of all of them in the same characterisation, possibly dynamically. We may then have to work harder to prove the correctness and completeness. This opportunity is denied to us by the object language translation approach. We have not performed such studies to date, but believe that the presentation of this chapter, with the motivation provided by the analysis of the sequent calculus, will serve as a good starting point for such studies.

Chapter 10

Conclusions.

10.1 Summary of results.

The research reported in this thesis concerned the automation of proof search within mathematical logics. It lies in the area of automated theorem proving (ATP). In this context, our main problem was to formulate efficient methods for automated proof search within an important class of non-classical logics comprising:

- the modal logics: K, K4, D, D4, T, S4 and S5, and
- intuitionistic logic.

These logics, and their derivatives, are in widespread use within Computing Science and Artificial Intelligence, mostly in applications that require efficient methods of proof search.

10.1.1 Background for the solution.

In Part I of this thesis we presented a theoretical reconstruction of Bibel's Connection Calculus [Bib82a]: a matrix characterisation of validity for first-order classical logic. Our main contribution was the decomposition of the method into

a collection of theoretically motivated techniques for overcoming certain types of redundancy within the search spaces induced by sequent calculi.

Three classes of redundancy were identified:

- Notational redundancy: considerable duplication of the same information.
- Relevance: the inclusion in the search space of branches that cannot lead to a proof.
- Order dependence: the need to explore alternative branches in the search space that differ only in the order in which certain sequent rules are applied.

10.1.1.1 Techniques for removing notational redundancy.

The techniques isolated for the removal of notational redundancies involved a special use of the formation tree of a formula and an indexing technique for capturing the *genericity* of the quantifiers. This enables the representation of derivations in terms of the syntactic structure of the formula being tested for validity. The whole scheme is a theoretically motivated reconstruction of *structure sharing* [BM72]. We described how its application within sequent-based search relies on the system possessing the *subformula property.*

The isolation of this technique from the overall matrix framework supports its use to remove notational redundancies in the implementation of sequent or tableau-based proof systems for any logic, provided the proof system possesses the subformula property.

10.1.1.2 Techniques for removing redundancies of relevance.

The techniques isolated for the removal of problems of relevance were the notions of *path*, *polarity* and *connection.* It is the notion of path that makes the label "matrix" appropriate for the final characterisation and gives proof methods based on the characterisation their path-checking flavour. A path through

a formula, A, is defined so as to represent a potential leaf of a sequent derivation of the endsequent: $\longrightarrow A$. The set of paths through A represents the set of potential leaves of *any* sequent derivation of $\longrightarrow A$ (given a bound on the number of duplications of subformulae of A). A connection in the formula is a representation of the distinguished atomic S-formulae within a (potential) basic sequent. It consists of two atomic formula occurrences with the same predicate symbol and of differing polarities (*i.e.*, a positive and a negative occurrence of a given proposition). If a path contains a connection it has the potential to represent a closed leaf of a derivation. To overcome problems of relevance we search for connections directly rather than adopt the standard connective-based approach. That is, we search directly amongst the potentially closed leaves for a subset suitable for the formation of a proof of the endsequent.

The effect of this is to replace an indirect search for basic sequents with a directed one. This eliminates from the (direct) matrix search space those parts of the (indirect) sequent search space that are irrelevant for the construction of basic sequents and hence proofs. If every path through A contains one of a given set of connections the set is said to *span* the formula. The existence of a spanning set of connections for A entails the existence of a sequent proof of $\longrightarrow A$ (and vice versa), and therefore the classical validity of A (and vice versa).

The notion of a connection arises from the nature of the basic sequent of the calculus:

$$\Gamma, A \longrightarrow A, \Delta$$

which in turn arises from the reflexivity of the consequence relation for the logic: $A \models A$. Such a property is practically taken to be a defining characteristic of a consequence relation ("A follows from A") and hence is very likely to hold for any logic of interest.

10.1.1.3 Techniques for the removal of order dependence.

For first-order logic the existence of a spanning set of connections is not sufficient to characterise validity since the quantifier rules induce constraints on the way in

which derivations are constructed. The sequents resulting from the application of the same two rules to the same two (indpendent) S-formulae can differ depending on the order in which the rules are applied. We called this a problem of the "order dependence" of certain sequent rules.

The techniques isolated for the removal of this problem were the notions of *complementarity* and the *admissibility* of mappings between (constructs representing) certain subformulae of the endsequent $\longrightarrow A$. The subformulae of significance are the side-formulae of the order dependent rules of the calculus. For classical logic these were shown to be the immediate subformulae of quantified subformulae of A. The mapping represents the coherence of the choice of parameters for the free variables of (free) atomic subformulae in a derivation. Such coherence is required so that the two atomic components of a connection can be construed as the distinguished antecedent and succedent formulae of a basic sequent (*i.e.*, they must be identical as formulae). We showed how such a mapping induces a *reduction ordering*: a transitive relation over subformulae of the endsequent that we are proving. The reduction ordering represents the constraints on the order in which immediate subformulae of quantified (sub)formulae may be introduced as S-formulae into a derivation. The constraints arise from the provisos on the two "existential" rules: $\longrightarrow \forall$ and $\exists \longrightarrow$. *Admissible* mappings are those whose reduction orderings are irreflexive.

A connection is defined to be *complementary* under an admissible mapping just in case its atomic components are identical under the mapping. The admissibility condition ensures that at least one sequent derivation exists in which the required coherence in the choice of parameters is realised so that the current set of connections form the closed leaves of the derivation. A spanning set of connections in A, complementary under some such admissible mapping, thus entails the existence of a sequent proof of $\longrightarrow A$ (and vice versa), and hence the (first-order) validity of A (and vice versa). Robinson's unification algorithm (or more efficient refinements of it) can be used to compute the appropriate mappings. Unification is used to ensure the *existence* of a correct order of sequent rule applications to produce a proof of the formula. No single concrete order

need be preferred. This technique removes the order dependence, induced in the sequent search space by the quantifier rules, from the matrix search space.

The definitions of admissibility and of complementarity depend on the nature of the sequent system.

10.1.1.4 Conclusions.

Whilst the basic nature of this matrix characterisation of validity in classical logic is due to Bibel, we believe our *technical* formulation of the individual techniques to be quite significant. For example, the view we have developed of the utility of unification for overcoming problems of order dependence permits the use of unification in new ways for the automation of proof search in mathematical logics. A good indication of this potential is provided by our treatment of modal and intuitionistic logics summarised below. We utilised Smullyan's uniform notation [Smu68] extensively and reformulated the notions of multiplicity and path to relate them more closely with sequent-based ideas.

In summary: our contribution has been to identify powerful techniques within Bibel's Connection Calculus [Bib80,Bib82c] for classical logic for improving the efficiency of sequent-based proof procedures in general. We have abstracted these techniques from a dependence on the details of classical logic by a proof-theoretic analysis of their effect on the search space induced by a cut-free sequent calculus for that logic.

Once identified, these techniques can be applied in new ways when and where their prerequisite conditions apply. In particular we used them in this thesis to develop efficient methods of proof search in non-classical logics. The results of these applications are summarised in the next two subsections.

10.1.2 Matrix proof methods for modal logics.

In Part II of the thesis we developed matrix characterisations of validity for the modal logics K, K4, D, D4, T, S4 and S5 in their varying, cumulative and

constant domain versions, a total of 20 distinct first-order modal logics in all. (The constant and cumulative domain versions of S5 are equivalent.)

The solution is motivated by a discussion of redundancies in the proof search space induced by cut-free sequent calculi for the modal logics. We note that cut-free sequent systems do not exist for all of the logics considered. This fact does not present a problem in the uniform application of techniques for the removal of the redundancies identified.

By judicious use of the techniques isolated in Part I of this thesis and summarised above, we succeeded in retaining the basic structure of the matrix characterisation for classical logic. Validity checking is reduced to a process of path-checking and complementarity tests for connections. As a consequence, search methods developed for use with the classical matrix characterisation (*eg.*, [Bib77, Bib82b,HB82]) are applicable *without change* to the modal logics. We have thus succeeded in extending one of the most efficient proof methods developed for classical logic (see [Bib82b]) to this important class of non-classical logics without compromising the basic computational properties of the method.

The main technical problem solved in order to achieve these results was the removal of problems of order dependence induced by the modal operators. In terms of the techniques summarised in the previous section the subformulae that are of significance are the side formulae of the modal rules, *i.e.*, the immediate subformulae of modally quantified subformulae of the endsequent: $\longrightarrow A$. We developed representations of these subformulae using the notion of a *prefix* adapted from Fitting's systems of tableaux [Fit72,Fit83] and Kanger's "spotted" sequent system for S5 [Kan57]. Each atomic subformula of A receives a prefix representing the modal context in which it appears. Mappings are defined over these prefixes. Connections are defined to be complementary just when an admissible mapping identifies the prefixes of the atomic formulae of the connection. Such a mapping induces a reduction ordering in the same manner as the corresponding mapping in the classical case summarised above. Once again admissible mappings can be computed by means of unification algorithms. We return to this point shortly.

The definition of admissibility is the key component of the treatment. It comprises two central conditions: the first (and logic-dependent) condition stipulates that the mapping respects a so-called *accessibility* relation on prefixes. This relation reflects basic properties of the semantics of the logic. The second (and logic-independent) condition stipulates that the reduction ordering induced by the mapping is irreflexive. These definitions are so arranged that the existence of a spanning set of complementary connections ensures the validity of A in the particular modal logic (and vice versa).

10.1.2.1 First-order modal logics.

The combination of the solution to specifically modal problems summarised above and the solution to the problems concerning quantifiers is straightforward. The first-order and modal mappings are independent. Both induce a reduction ordering. The reduction ordering for the quantified modal systems is the union (and transitive closure) of the two separate orderings.

Constant domains. In the simplest case of the constant domain variants (where there is no semantic correlation between modality and existence) the logic-dependent condition on the modal mapping is retained.

Varying and cumulative domains. For these variants, where there is a correlation between modality and existence, the modal and first-order mappings interact. The first-order mapping must respect the modal mapping in a complex way. Positions identified by the first-order mapping must have identical prefixes (varying domains), or prefixes which are accessible from each other (cumulative domains), under the modal mapping. The essence of this relationship is taken from Fitting[Fit83] and his prefixed tableau systems. We adapt it to the matrix systems and use unification to remove the order dependence inherent in his tableau systems.

10.1.2.2 The use of unification.

We mentioned that unification algorithms could be used to compute the modal mappings. In fact, we are interested in the computation of mappings that satisfy the logic-dependent condition, *i.e.*, they respect the accessibility relation on prefixes for a given logic. We demonstrated that such mappings can be calculated by means of standard unification algorithms. The most complex algorithm needed is a restricted version of string unification [Sie75].

10.1.2.3 Decision procedures.

We also outline how the theoretical basis of the matrix characterisations can be used to formulate efficient decision procedures for the propositional fragments of the modal logics. We develop such a procedure for S5 explicitly and indicate how it may be generalised to the other modal logics considered.

10.1.2.4 Related work.

Finally, we use the analytic tools developed in this thesis to classify the redundancies in the search spaces of the major proposals in the literature for efficient proof systems for similar classes of modal logic considered here. We identify three types of system:

- Proof systems based on sequent/tableau calculi.
- Systems based on either clausal or non-clausal resolution.
- Hybrid proof systems based on a mixture of sequent/tableau ideas and resolution.

To summarise our findings:

- The spaces generated by sequent/tableau proof systems, in general contain all three types of redundancy: notational, relevance and order dependence.

- The spaces generated by the resolution and hybrid tableau/resolution proof systems:

  - can be extended to utilise structure sharing methods to overcome the notational problems;
  - overcome the relevance problem by means of connections,
  - overcome the order problems of the quantifiers using unification, but
  - *fail* to deal with the order problems associated with modalities.

  Additionally, none of the resolution proposals capture the full range of modal logics treated here, though in some cases the systems can be extended.

We show that each of the proposals reviewed provide a more redundant basis for automated proof search than the modal matrix characterisations summarised above. None of the proposals overcome the order dependence of the modal rules, or more abstractly, the interaction of modalities. We point to our use of unification to solve this problem as a central contribution.

10.1.2.5 Conclusions.

We have succeeded in extending one of the most efficient proof methods developed for classical logic to an important class of non-classical logics without compromising the basic computational properties of the method. The solution is both effective and comprehensive. We believe these results to be a major contribution to the field of automated theorem proving.

10.1.3 A matrix proof method for intuitionistic logic.

The modal logics treated in this thesis are *extensions* of classical logic. That is to say, they contain first-order classical logic as a subsystem. In Part III of the thesis we reinforced the results of Parts I and II that are summarised above by

developing a matrix characterisation of validity for first-order intuitionistic logic. Intuitionistic logic is a *subsystem* of classical logic, and is perhaps the archetypal "non-classical" logic. This result achieves two objectives:

1. It provides an efficient proof method for what is a logic of considerable current interest.

2. It provides further evidence of the power of the approach developed in this thesis for the efficient automation of proof search in non-classical logics.

We obtained this result in our standard way. By analysing the redundancies within the search space induced by a standard cut-free sequent calculus for the logic we were able to adapt the techniques isolated in Part I to their removal. As was the case for the modal logics, the main technical problem was the removal of problems of order dependence. This time it was the sentential connectives rather than additional operators that were the root cause of the problem. We adopted a prefix technique again inspired by an embedding of intuitionistic logic in S4 modal logic. The correctness and completeness of the characterisation was established by this root also.

10.2 Implications and Future work.

We have succeeded in our goal of formulating efficient methods for automated proof search within an important class of non-classical logics. The solution is both comprehensive and effective. We have compared our methods with others proposed in the literature and demonstrated the advantages of the matrix methods.

In developing this solution we have achieved what could be a more significant result. We have isolated powerful techniques for improving the efficiency of sequent-based proof procedures in general by the analysis of an existing matrix proof method for classical logic. These techniques can be applied individually when and where the prerequisite conditions apply. In particular they can be used

to develop efficient methods of proof search in other non-classical logics. We believe these results in themselves to be a significant contribution to understanding in the field of Automated Theorem Proving.

In this penultimate section we evaluate what we have achieved with an eye to directions for future research.

10.2.1 A more abstract approach.

We argued in the introduction to this thesis that non-classical logics are the rule rather than the exception. Representation and reasoning within a domain requires a language with a well-defined semantic structure that reflects general properties of that domain. As new domains are encountered so will new logics be developed. We believe that in applications in Computing Science and Artificial Intelligence involving the use of logic to represent and manipulate information, "logic" is a synonym for "general theory." This places a heavy burden on the designer of proof systems. The day after one's latest success in taming a weird logic for application A, someone will suggest that you have a go at their new formalism for application B. Adhoc solutions may solve individual practical problems but they do not yield robust theories with which to tackle new problems.

Although the central problem addressed in this thesis is the automation of particular logics of current interest, we have endeavoured to maintain as abstract a view of our solution methods as is technically feasible given our current understanding. In Part I of the thesis we managed to abstract certain techniques from their original setting (embedded within Bibel's Connection Calculus) and give them a certain logic-independent flavour. This level of abstraction was used to good effect in Parts II and III.

We would have liked to have been able to present a single matrix framework — a generic Connection Calculus — and then described how to specialise it to the logics considered in this thesis. Our understanding is not yet deep enough to achieve this level of abstraction. This is a major area for further research.

There are a number of themes woven throughout the thesis motivating our discussion. We would like to be able to formalise these intuitions. One such theme is the correlation between the existence of a cut-free sequent calculus for a logic (containing a standard basic sequent and possessing the subformula property) with the existence of a matrix characterisation of validity in the logic. Is the former a sufficient condition for the latter? A satisfactory treatment of this question may be a prerequisite for the formulation of a generic matrix method.

On the other hand the reliance on proof-theoretic is: ıay be slightly misleading. Notice that we argued explicitly (following Fine [Fin79]) that cut-free sequent calculi for S5 and the constant domain variants of the first-order modal logics were not feasible. Nevertheless we succeeded in formulating simple matrix methods for these logics. The proof-theoretic notions on which we have relied are merely visible symptoms of a more abstract structure. We note that all the logics considered in this thesis admit a form of Kripke semantics. The formalisation of the matrix techniques in terms of the properties of such semantic bases is an area of further research we would definitely like to pursue. In [Smu70], Smullyan captures commonalities between classical, modal and intuitionistic logics quite uniformly in terms of his systems of analytic tableau (equivalent to cut-free sequent systems). This work may provide a suitable starting point for the research outlined above.

The question arises as to how flexible the matrix methods are. Some domains will need combinations of logics that deal with different aspects of the domain. Are the matrix methods applicable to such hybrid logics? It is quite possible that the answer is "no." The matrix methods give a global characterisation of validity involving all the connectives/operators/quantifiers of the logic. A certain amount of coherence is required. This is usually the case with *analytic* proof systems such as analytic tableau and cut-free sequent calculi, and this property may be the key to explaining why such methods support matrix-type refinements. The matrix methods will most likely not be applicable to "ill-structured" logics where many operators are considered in an adhoc manner. Abadi and Manna's

axiomatic techniques [AM86a,AM86b] are more suited to these logics, but at a cost of efficiency. This should probably be seen as inevitable.

One further possibility is to try to capture the relationship between the matrix methods developed here and the results of embeddings in classical logic. Prefixes were initially interpreted proof-theoretically, but we resorted to semantic justifications in the end. In the justifications prefixes were interpreted as the names of points in a potential falsifying model. This could be made explicit by representing the semantics of the (modal) logic in, say, classical logic. Our use of unification would then reduce to a method for dealing with the special theory of the accessibility relation for the logic. This is an avenue worth exploring because it would enable the use of existing proof procedures for classical logic in non-classical logics.

As a caveat here: we believe that characterising a logic directly can lead to many benefits. The decision procedures outlined in Chapter 7 could probably not been formulated outwith the matrix framework. We defined the notion of an $\mathcal{L}$-Hintikka multiplicity which set effective bounds on the size of the space to be searched in order to conclude that a formula is falsifiable. We indicated how this result could be extended to the other logics. Some technical work on this need to be done. It would also be interesting to define the notion of *minimal* $\mathcal{L}$-Hintikka multiplicities: multiplicities that are provably the smallest possible for a given logic.

Another example of how a detailed analysis of a logic rather than its embedding in a more general setting such as classical logic can yield dividends is the possibility of formulating powerful notions of purity in non-classical logics. We remarked on one such possibility in Chapter 7 where the K-conditions on admissibility rejected a potential connection. There is scope for exploiting the matrix characterisations of modal logic by building-in specific modal techniques to the improve the standard path-checking algorithms. Modal definitions of purity are but one avenue that could be pursued.

We have already extended the work reported in this thesis by developing a matrix characterisation for Ketonen and Weyrauch's Direct Predicate Calculus

[KW84]: a decidable subsystem of classical logic defined by removing the contraction rule from a cut-free sequent calculus. This suggests it may be possible to capture relevance logics [AB75] which require similar care in the duplication of formula in derivations. We have not looked at this question in any detail to-date.

Andrews and his co-workers have developed matrix systems for classical higher-order logic (Church's Type Theory) [Mil84,Pfe84], it would be interesting to try to combine their results with the method developed here for first-order intuitionistic logic and develop efficient methods of proof search for logics such as intuitionistic type theory [Mar82,Con86].

10.3 Summary of the thesis.

The research reported in this thesis is concerned with the automation of proof search within mathematical logics. It lies in the area of automated theorem proving (ATP). In this context, our main problem was to formulate efficient methods for automated proof search within an important class of non-classical logics comprising:

- the modal logics: K, K4, D, D4, T, S4 and S5, and
- intuitionistic logic.

These logics, and their derivatives, are in widespread use within Computing Science and Artificial Intelligence, mostly in applications that require efficient methods of proof search.

Our contribution is two-fold,

- We have succeeded in formulating efficient matrix proof methods for the target logics, and have therefore solved our main problem.

- In the pursuit of this solution we have isolated powerful techniques for improving the efficiency of sequent-based proof procedures in general. These techniques can be applied individually when and where the prerequisite conditions apply. In particular they can be applied to develop efficient methods of proof search in other non-classical logics.

We believe that this research is an important contribution to the field of automated theorem proving.

Bibliography

[AB75] A.R. Anderson and B.D. Belnap. *Entailment.* Volume 1, Princeton University Press, Princeton, NJ, 1975.

[AM86a] M. Abadi and Z. Manna. Modal theorem proving. In J.H. Siekmann, editor, *8th International Conference on Automated Deduction*, pages 172–189, July 1986. Lecture Notes in Computer Science, Volume 230, Springer Verlag.

[AM86b] M. Abadi and Z. Manna. A timely resolution. In *Proceedings of Symposium on Logic in Computer Science*, pages 176–186, June 1986.

[And81] P.B. Andrews. Theorem-proving via general matings. *Journal of the Association for Computing Machinery*, 28(2):193–214, April 1981.

[Bib77] W. Bibel. *Tautology Testing with an improved Matrix Reduction.* Research Report TUM-INFO-7706, Institut fur Informatik, Technische Universität München, May 1977.

[Bib80] W. Bibel. *The Complete Theoretical Basis for the Systematic Proof Method.* Bericht ATP-6-XII-80, Technische Universität München, December 1980.

[Bib81] W. Bibel. On matrices with connections. *Journal of the Association for Computing Machinery*, 28(4):633–645, October 1981.

[Bib82a] W. Bibel. *Automated Theorem Proving.* Friedr. Vieweg & Sohn, Braunschweig, 1982.

[Bib82b] W. Bibel. A comparative study of several proof procedures. *Artificial Intelligence*, 18:269–293, 1982.

[Bib82c] W. Bibel. Computationally improved versions of Herbrand's Theorem. In J. Stern, editor, *Proceedings of the Herbrand Symposium, Logic Colloquium '81*, pages 11–28, North-Holland Publishing Co., 1982.

[Bib83] W. Bibel. Matings in matrices. *Communications of the ACM*, 28(4):844–852, November 1983.

[BM72] R.S. Boyer and J.S. Moore. The sharing of structure in theorem-proving programs. In B. Meltzer and D. Michie, editors, *Machine Intelligence 7*, pages 101–116, Edinburgh University Press, 1972.

[Bro75] L.E.J. Brouwer. *Collected Works.* Volume 1, North Holland, 1975.

[BT75] W.W. Bledsoe and M. Tyson. *The UT Interactive Prover.* Research Report ATP-17, Departments of Mathematics and Computer Sciences, University of Texas at Austin, May 1975.

[CH85] Th. Coquand and G. Huet. Constructions: a higher order proof system for mechanising mathematics. In B. Buchberger, editor, *EUROCAL '85: European Conference on Computer Algebra*, pages 151–184, Springer Verlag, 1985.

[CK73] C.C. Chang and H.J. Keisler. *Model Theory.* Volume 73 of *Studies in Logic*, North Holland, 1973.

[CL73] C-L. Chang and R.C-T. Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, 1973.

[Con86] R.L. Constable *et al. Implementing Mathematics with the Nuprl Proof Development System.* Prentice Hall, 1986.

[D84] Schmidt D. A programming notation for tactical reasoning. In Shostak R.E., editor, *7th International Conference on Automated Deduction*, pages 445–459, May 1984. Lecture Notes in Computer Science, Volume 170, Springer Verlag.

[Dum77] M. Dummett. *Elements of Intuitionism.* Oxford University Press, 1977.

[DWP81] D.R. Dowty, R.E. Wall, and S. Peters. *Introduction to Montague semantics.* Reidel, Dordrecht, 1981.

[Far82] L. Fariñas-del-Cerro. A simple deduction method for modal logic. *Information Processing Letters*, 14(2), 1982.

[Far83] L. Fariñas-del-Cerro. Temporal reasoning and termination of programs. In S. Amarel, editor, *8th International Joint Conference on Artificial Intelligence*, pages 926–929, 1983.

[Far86] L. Fariñas-del-Cerro. Resolution modal logics. *Logique et Analyse*, 110-111:153–172, 1986.

[Fin79] K. Fine. Failures of the interpolation lemma in quantified modal logic. *JSL*, 44(2), June 1979.

[Fit69] M.C. Fitting. *Intuitionistic logic, model theory and forcing. Studies in logic and the foundations of mathematics*, North Holland, 1969.

[Fit72] M.C. Fitting. Tableau methods of proof for modal logics. *Notre Dame Journal of Formal Logic*, XIII:237–247, 1972.

[Fit83] M.C. Fitting. *Proof methods for modal and intuitionistic logics.* Volume 169 of *Synthese library*, D. Reidel, Dordrecht, Holland, 1983.

[G69] Gentzen G. Investigations into logical deduction. In Szabo M.E., editor, *The collected papers of Gerhard Gentzen*, chapter 3, pages 68–131, North-Holland, Amsterdam, 1969. Translation.

[Gab85] D.M. Gabbay. N-prolog: an extension of prolog with hypothetical implication. ii: logical foundations and negation as failure. *Logic Programming*, 2(4):251–283, 1985.

[Gal86] J.H. Gallier. *Logic for Computer Science: Foundations of Automated Theorem Proving.* Volume 5 of *Computer Science and Technology*, Harper & Row, 1986.

[GMW79] M.J.C. Gordon, A.J. Milner, and C.P. Wadsworth. *Edinburgh LCF - A mechanised logic of computation.* Volume 78 of *Lecture Notes in Computer Science*, Springer Verlag, 1979.

[God69] K. Gödel. An interpretation of the intuitionistic sentential logic. In J. Hintikka, editor, *The Philosophy of mathematics*, pages 128–129, OUP, 1969.

[Gra84] P.M.D. Gray. *Logic, Algebra and Databases.* Volume 29 of *Computers and their applications*, Ellis Horwood, 1984.

[Har79] D. Harel. *First-Order Dynamic Logic.* Volume 68 of *Lecture Notes in Computer Science*, Springer Verlag, 1979.

[HB82] K.M. Hörnig and W. Bibel. Improvements of a tautology testing algorithm. In D.W. Loveland, editor, *6th International Conference on Automated Deduction*, pages 326–342, Springer Verlag, 1982. Lecture Notes in Computer Science No. 138.

[HC68] G.E. Hughes and M.J. Cresswell. *An Introduction to Modal Logic.* Methuen, London, 1968.

[HM84] J.Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. In *3rd ACM Conference on the Principles of Distributed Computing*, pages 50–61, 1984.

[HM85] J.Y. Halpern and Y. Moses. A guide to the modal logics of knowledge and belief:preliminary draft. In *9th International Joint Conference on Artificial Intelligence*, pages 479–490, 1985.

[JR87] P. Jackson and H. Reichgelt. A general proof method for first-order modal logic. In J. McDermott, editor, *10th International Joint Conference on Artificial Intelligence*, pages 942–944, Morgan Kaufmann Inc., 1987.

[Kan57] S. Kanger. *Provability in logic.* Volume 1 of *Stockholm Studies in Philosophy*, Almqvist and Wiksell, Stockholm, 1957.

[KK71] R. Kowalski and D. Kuehner. Linear resolution with selection function. *Artificial Intelligence*, 2:227–260, 1971.

[Kle68] S.C. Kleene. *Mathematical Logic.* John Wiley & Sons, 1968.

[Kon84] K. Konolige. *A Deduction Model of Belief and its Logics.* PhD thesis, Stanford University, 1984.

[Kon86] K. Konolige. Resolution and quantified epistemic logics. In J.H. Siekmann, editor, *8th International Conference on Automated Deduction*, pages 199–208, July 1986. Lecture Notes in Computer Science, Volume 230, Springer Verlag.

[Kow75] R. Kowalski. A proof procedure using connection graphs. *Journal of the Association for Computing Machinery*, 22(4):572–595, 1975.

[Kow79] R. Kowalski. *Logic for Problem Solving. Artificial Intelligence Series*, North Holland, 1979.

[Kri63] S.A. Kripke. Semantical analysis of modal logic I, normal propositional calculi. *ZML*, 8:67–96, 1963.

[KW84] J. Ketonen and R. Weyhrauch. A decidable fragment of predicate calculus. *Theoretical Computer Science*, 32(3):297–309, 1984.

[Lyn66] R.C Lyndon. *Notes on Logic.* Van-Nostrand, Princeton, 1966.

[Mar82] P. Martin-Löf. *Constructive mathematics and computer programming*, pages 153–175. Volume IV of *Logic, Methodology and Philosophy of Science*, North-Holland, Amsterdam, 1982.

[McC86] L.T. McCarty. *Fixed point semantics and tableau proof procedures for a clausal intuitionistic logic.* Technical Report LRP-TR-18, Department of Computer Science, Rutgers, 1986.

[Mel71] B. Meltzer. Prolegomena to a theory of efficiency of proof procedures. In *Artificial Intelligence and Heuristic Programming*, pages 15–33, Edinburgh University Press, 1971.

[Mil84] D.A. Miller. Expansion tree proofs and their conversion to natural deduction proofs. In R.E. Shostak, editor, *7th International Conference on Automated Deduction*, pages 375–393, May 1984. Lecture Notes in Computer Science, Volume 170, Springer Verlag.

[Min70] G.E. Mints. Cut-free calculi of the S5 type. In *Studies in constructive mathematics and mathematical logic, Part II, Seminars in Mathematics*, pages 115–120, 1970.

[Moo80] R.C. Moore. *Reasoning about knowledge and action.* Technical Note 191, SRI International, Menlo Park, Ca., 1980.

[Mur82] N.V. Murray. Completely non-clausal theorem proving. *Artificial Intelligence*, 18:67–85, 1982.

[MW80] Z. Manna and R. Waldinger. A deductive approach to program synthesis. *ACM Transactions on Programming Languages and Systems*, 2(1):90–121, 1980.

[OS86] F. Oppacher and E. Suen. Controlling deduction with proof condensation and heuristics. In J.H. Siekmann, editor, *8th International*

Conference on Automated Deduction, pages 384–393, July 1986. Lecture Notes in Computer Science, Volume 230, Springer Verlag.

[Pfe84] F. Pfenning. Analytic and non-analytic proofs. In R.E. Shostak, editor, *7th International Conference on Automated Deduction*, pages 375–393, May 1984. Lecture Notes in Computer Science, Volume 170, Springer Verlag.

[Plo72] G. Plotkin. Building in equational theories. In *Machine Intelligence*, Edinburgh University Press, 1972.

[Pne77] A. Pneuli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*, pages 46–57, 1977.

[Pra60] D. Prawitz. An improved proof procedure. *Theoria*, 26:102–139, 1960.

[Rob65] J.A. Robinson. A machine oriented logic based on the resolution principle. *J Assoc. Comput. Mach.*, 12:23–41, 1965.

[Sat77] M. Sato. A study of kripke-type models for some modal logics by gentzen's sequential method. In *Publications*, pages 381–468, Research Institute for Mathematical Sciences, Kyoto University, 1977.

[Sie75] J. Siekmann. *String Unification.* Memo CSM-7, Essex University, 1975.

[Sie82] J. Siekmann. A noetherian and confluent rewrite system for idempotent semigroups. In *Semigroup Forum*, 1982.

[Sie84] J.H. Siekmann. Universal unification. In Shostak R.E., editor, *7th International Conference on Automated Deduction*, pages 1–42, May 1984. Lecture Notes in Computer Science, Volume 170, Springer Verlag.

[Smu68] R.M. Smullyan. *First-Order Logic.* Volume 43 of *Ergebnisse der Mathematik*, Springer-Verlag, Berlin, 1968.

[Smu70] R.M. Smullyan. Abstract quantification theory. In J. Myhill and R.E. Vesley, editors, *Intuitionism and Proof Theory*, pages 79–91, North Holland, Amsterdam, 1970.

[Sti85a] M.E. Stickel. Automated deduction by theory resolution. *Journal of Automated Reasoning*, 1:333–355, 1985.

[Sti85b] C. Stirling. *Modal logics for communicating systems.* Internal Report CSR-193-85, Dept. of Computer Science, Edinburgh University, 1985.

[Tur84] R Turner. *Logics for Artificial Intelligence.* Ellis Horwood, 1984.

[Wal86] L.A. Wallen. Generating connection calculi from tableau- and sequent-based proof systems. In A.G. Cohn and J.R. Thomas, editors, *Artificial Intelligence and its Applications*, pages 35–50, John Wiley & Sons, 1986. Proceedings of AISB85, Warwick, England, April 1985.

[Wal87] L.A. Wallen. Matrix proof methods for modal logics. In J. McDermott, editor, *10th International Joint Conference on Artificial Intelligence*, pages 917–923, Morgan Kaufmann Inc., 1987.

[Wil86] G.V. Wilson. *Implementation of a connection method theorem-prover for S5 modal logic.* Master's thesis, Department of Artificial Intelligence, University of Edinburgh, 1986.

[Wri85] Graham Wrightson. Nonclassical logic theorem proving. *Journal of Automated Reasoning*, 1(1):35–37, 1985.

[WW87] L.A. Wallen and G.V. Wilson. A computationally efficient proof system for S5 modal logic. In J. Hallam and C. Mellish, editors, *Advances*

in Artificial Intelligence, pages 141–153, John Wiley & Sons, 1987. Proceedings of AISB87, Edinburgh, Scotland, April 1987.

GENERATING CONNECTION CALCULI FROM TABLEAU AND SEQUENT BASED PROOF SYSTEMS*

Lincoln A. Wallen Edinburgh University, Edinburgh EH1 2QL, U.K.

ABSTRACT

The relationship between Bibel's *connection calculus* and Smullyan's system of *analytic tableaux* for first-order classical logic is discussed in depth. The major features of this analysis form a methodology for constructing a connection calculus for a given logic from a tableau or sequent based proof system for that logic.

It is claimed that the methodology is sufficiently general to be applied to a number of common first-order logics. This claim is supported by a brief description of the results of its application to a sequent calculus for S_5 modal logic.

* This research was support in part by SERC grants GR/C/35967, GRB/67766, and an SERC studentship to the author.

Artificial Intelligence and its Applications, edited by A. G. Cohn and J. R. Thomas

1. INTRODUCTION

For many years implementations of Robinson's resolution system (Robinson, 1965), and refinements thereof, have been the dominant means of automating deduction in first-order classical logic. Recently Bibel (1981), and independently Andrews (1981), have developed a more subtle basis for deduction in this logic based on the notions of *paths* and *connections*. The efficiency of proof procedures developed within this framework compares very favourably with all common refinements of resolution (Bibel, 1982a).

More important however is the relationship Bibel's *connection calculus* bears to various techniques based on Gentzen's sequent calculus (Gentzen, 1969) and in particular its relationship to Beth's method of *semantic tableaux* (Beth, 1959) as modified by Smullyan (1968). Tableau and Gentzen systems have become a standard metamathematical tool for the specification and investigation of different logics in both mathematics and computer science, e.g. Fitting (1983), Kanger (1957), Nishimura (1983). Indeed, using his systems of *analytic tableaux*, Smullyan (1970) has developed a notion of an abstract quantification theory; a unifying framework for many first-order logics.

This paper sets out in some detail the relationship between the connection calculus and Smullyan's system; first for (classical) propositional logic (§3–§6), and then for a first-order system (§7–§8). The relationship is developed by applying a series of transformations to the tableau system to derive the major features of the corresponding connection calculus.

I claim that this analysis forms the basis for a general methodology for the *efficient* automation of first-order logics specified using tableau or Gentzen systems. A similar analysis of Andrews' results (Andrews, 1981) may lead to an extension of the methodology to higher-order logics.

The methodology has been used successfully to develop a connection calculus for S_5 modal logic from Kanger's Gentzen-style system (Kanger, 1957). A brief description of this enterprise is contained in the penultimate section. The full details of the efficient modal proof system thus obtained will be reported elsewhere.

2. PRELIMINARIES

In the next few sections, while dealing with propositional logic, I shall use a language comprising a denumerable number of *propositional variables* P, Q, R,... together with the sentential connectives $\neg$, $\vee$, $\wedge$ and $\supset$ denoting negation, disjunction, conjunction and implication respectively. The *formulas* X, Y, Z,... of propositional logic are defined as usual. The propositional variables will sometimes be referred to as the *atomic* formulas.

For the connection calculus, where we require more structure on formulas, I shall make use of Bibel's notion of a *formula-tree*. An example of the formula-tree for the formula X, of the form $((P \vee Q) \wedge (P \supset Q)) \supset Q$, is shown in Fig. 1.

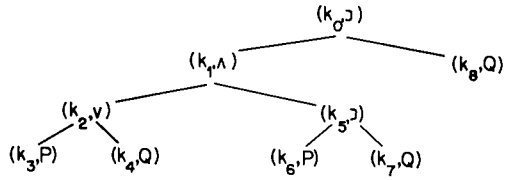


Figure 1: Formula-tree for $((P \vee Q) \wedge (P \supset Q)) \supset Q$

W_X denotes the set of *positions* $k_0 - k_8$ of X. The *label* λk of a position $k \in W_X$ is the piece of concrete syntax paired with it. So, for instance, $\lambda k_0 = \supset$, and $\lambda k_3 = \lambda k_6 = P$. The *tree-ordering* $<_X$ over W_X is the vertical partial order in the figure; e.g. $k_0 < xk_2$, and k_4 and k_6 are incomparable w.r.t. $<_X$. Henceforth the subscript X will be dropped if no confusion can arise. Finally let X_k denote the subformula of X rooted at position k; for example, $X_{k5} = P \supset Q$.

The following eight facts, derived directly from the semantic clauses for the sentential connectives, form the basis for both the method of analytic tableaux and the connection method. Under any interpretation

F1. For negation,
  (a) if $\neg$ Y is false, Y is true,
  (b) if $\neg$ Y is true, Y is false.

F2. For disjunction,
  (a) if $Y \vee Z$ is false, both Y and Z are false,
  (b) if $Y \vee Z$ is true, either Y or Z is true.

F3. For conjunction,
  (a) if $Y \wedge Z$ is false, either Y or Z is false,
  (b) if $Y \wedge Z$ is true, both Y and Z are true.

F4. For implication,
  (a) if $Y \supset Z$ is false, Y is true and Z is false,
  (b) if $Y \supset Z$ is true, either Y is false or Z is true.

3. TABLEAUX FOR PROPOSITIONAL LOGIC

This section contains a review of the basic structures underlying the method of analytic tableaux. The definitions that follow are taken, with slight modifications, from Smullyan (1968).*

We begin by defining the notion of a *signed* formula.

Definition 3.1: A *signed* formula is an expression TX or FX, where X is an unsigned formula. Informally TX and FX are read 'X is true' and 'X is false' respectively. The *conjugate* of a signed formula TX is the signed formula FX (and vice versa).

* In fact we use what Smullyan calls 'block tableaux' since the relationship these notational variants of analytic tableaux bear to the connection calculus and sequent calculi in general is quite direct.

Definition 3.2: An *analytic tableau* $\mathcal{T}$ for a formula X is an ordered dyadic tree, whose nodes are sets of occurrences of signed formulas, constructed as follows.

T0. The tree comprising a single node with {FX} at its root is a tableau for X. We call this unique tableau the *initial* tableau for X.

T1. $\mathcal{T}$ is a tableau for X just in case there exists a finite sequence $(\mathcal{T}_1, \ldots, \mathcal{T}_n = \mathcal{T})$ of ordered dyadic trees whose nodes are sets of occurrences of formulas, such that $\mathcal{T}_1$ is the initial tableau for X and for each $i < n$, $\mathcal{T}_{i+1}$ is a *direct extension* of $\mathcal{T}_i$. $\mathcal{T}$ is said to be an *extension* of $\mathcal{T}_1$.

A *direct extension* of a tableau is an application of one of the following rule schemata to an end-point S of one of its branches.

$$\frac{(S[F\neg Y])}{(S,TY)} \qquad \frac{(S[FY \vee Z])}{(S,FY,FZ)} \qquad \frac{(S[FY \wedge Z])}{(S,FY) \mid (S,FZ)} \qquad \frac{(S[FY \supset Z])}{(S,TY,FZ)}$$

$$\frac{(S[T\neg Y])}{(S,FY)} \qquad \frac{(S[TY \vee Z])}{(S,TY) \mid (S,TZ)} \qquad \frac{(S[TY \wedge Z])}{(S,TY,TZ)} \qquad \frac{(S[TY \supset Z])}{(S,FY) \mid (S,TZ)}$$

Of the two types of rule

$$\frac{(S[\alpha])}{(S, \alpha_1, \alpha_2)} \qquad \frac{(S[\beta])}{(S, \beta_1) \mid (S, \beta_2)}$$

the first kind should be read:

— to extend a brach with end-point S containing the formula α, adjoin the set $S \cup \{\alpha_1, \alpha_2\}$ as the sole successor of S;

and the second:

— to extend a branch with end-point S containing the formula β, adjoin $S \cup \{\beta_1\}$ and $S \cup \{\beta_2\}$ as the left and right successors respectively of S.

The formula α or β distinguished in an application of a rule is said to have been *reduced*.

Informally, the tableau method works by assuming X to be false (FX is placed at the root of the tableau) and attempting to show, using the eight facts presented in the previous section, that this assumption leads to a contradiction. Readers should convince themselves that the reduction rules for tableaux described above are merely a syntactic representation of those eight facts. The contradictions are the occurrences of two signed formulas TY and FY in the same node indicating that, under the assumptions pertaining for the construction of that branch, Y must be both true and false. The different branches arise because of the alternative conclusions possible from an assumption, say, that $Y \vee Z$ is true. We need to show that under either possibility (that Y is true, or that Z is true) we get a contradiction. Thus the method requires that there is a node containing such a contradiction on every branch. This motivates the following definitions.

Definition 3.3: An end-point S of a tableau is (*atomically*) *closed* just in case it contains a signed (atomic) formula and its conjugate, otherwise it is *open*. A tableau is (atomically) *closed* if the end-points of all its branches are (atomically) closed. By an (atomic) *proof* for a formula X we mean an (atomically) closed tableau for X.

A tableau proof of the tautology shown in figure 1 is displayed in figure 2.

Definition 3.4: An end-point S of a tableau is said to be *complete* just in case

$$\begin{aligned}
F\neg Y \in S &\Rightarrow TY \in S\\
T\neg Y \in S &\Rightarrow FY \in S\\
FY \vee Z \in S &\Rightarrow FY \in S \text{ and } FZ \in S\\
TY \vee Z \in S &\Rightarrow TY \in S \text{ or } TZ \in S\\
FY \wedge Z \in S &\Rightarrow FY \in S \text{ or } FZ \in S\\
TY \wedge Z \in S &\Rightarrow TY \in S \text{ and } TZ \in S\\
FY \supset Z \in S &\Rightarrow TY \in S \text{ and } FZ \in S\\
TY \supset Z \in S &\Rightarrow FY \in S \text{ or } TZ \in S
\end{aligned}$$

A tableau is said to be complete just in case the end-points of all its branches are complete.

A set of signed formulas which in addition to the above conditions does not contains a signed atomic formula and its conjugate, is sometimes called a *Hintikka*, or *downward saturated* set. Such sets are of importance due to the following fact proved in Smullyan (1968).

Fact 3.5: Every Hintikka set is satisfiable. In other words, any Hintikka set S may be completed to a model $\mathcal{M}$ in which

if FY ∈ S (resp. TY ∈ S), then Y is false (resp. true) in $\mathcal{M}$.

Thus, by definition

Fact 3.6: Any complete open end-point of any tableau is satisfiable.

We can obtain a complete tableau by ensuring that every signed formula in each end-point has been reduced once. If we manage to construct a branch with a complete open end-point S then we are guaranteed the existence of a model that satisfies S. In particular, since FX is a member of every node of any tableau for X, X must be false in this model. Hence X is not a tautology. Consistency follows directly from the eight facts

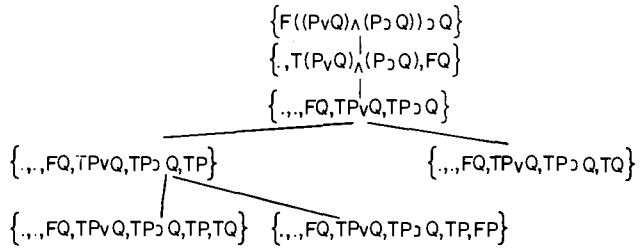


Figure 2: A tableau proof of $((P \vee Q) \wedge (P \supset Q)) \supset Q$

presented in the previous section. In conclusion then:

Theorem 3.7: X is a tautology of (classical) propositional logic if and only if every complete tableau for X is (atomically) closed.

We proceed to investigate the precise relationship between this theorem and a modified version of Bibel's *connection theorem* for propositional logic.

4. POLARITY

Consider again the reduction rules for tableaux. A moment's reflection should suffice to convince the reader that the sign with which a subformula of X can appear in *any* node of *any* tableau for X is uniquely determined by the structure of X.

This observation may be formalised by associating with each position $k \in W$ a *polarity* indicating the sign with which it will appear.

Definition 4.1: Suppose $n_0 \in \{0,1\}$. For a formula X, we define the *polarity* $\rho(k) \in \{0,1\}$ of a position $k \in W$ (with respect to n_0) inductively according to the structure of X.

Po0. If $X_k = X$, ie. $k = k_0$ the root position of X,

$\rho(k) = n_0$

Po1. If $X_k = \neg Y$ for some subformula $Y = X_r$,

$\rho(r) = (\rho(k) + 1) \bmod 2$.

Po2. If $X_k = Y \vee Z$ for some subformulas $Y = X_r$ and $Z = X_s$ of X,

$\rho(r) = \rho(s) = \rho(k)$.

Po3. If $X_k = Y \wedge Z$ for some subformulas $Y = X_r$ and $Z = X_s$ of X,

$\rho(r) = \rho(s) = \rho(k)$.

Po4. If $X_k = Y \supset Z$ for some subformulas $Y = X_r$ and $Z = X_s$ of X,

$\rho(r) = (\rho(k) + 1) \bmod 2$, and $\rho(s) = \rho(k)$.

Remark 4.2: ρ is well-defined. Give a polarity n_0 for the root position k_0 of a formula X, the above definition assigns a unique element of $\{0,1\}$ to each $k \in W$.

We have the following

Proposition 4.3: Let $\mathcal{T}$ be a tableau for X, and assign $\rho(k_0) = 0$ where k_0 is the root position of the formula-tree for X. For any subformula $Y = Y_k$ of X that appears signed on $\mathcal{T}$

Y appears as FY (resp. TY) if and only if $\rho(k) = 0$ (resp. 1).

Thus the notion of polarity fulfills the same purpose as Smullyan's notion of signed formulas. Our first modification to the tableaux method is to make the nodes of tableaux sets of positions instead of signed formulas. We can then reformulate the reduction rules for tableaux in terms of positions

and their polarity. Such a modification amounts to a form of *structure sharing* (Boyer and Moore, 1972) and is invaluable when implementing such proof procedures.*

The following definition completes this modification.

Definition 4.4: A *connection* in a formula X is an (unordered) pair $\{k_1, k_2\}$ of positions in X such that

C0. $\lambda k_1 = \lambda k_2 = P$ for some atomic formula P, and

C1. $\rho(k_1) \neq \rho(k_2)$.

Clearly the existence of a connection in a node of a modified tableau is equivalent to saying that the node of the corresponding analytic tableau is atomically closed.

5. PATHS

We can obtain a complete tableau for a formula X with a finite number of reductions since (in propositional logic at least) we are required to reduce each (non-atomic) signed formula in a node just once, in order that the end-points of the resulting branches be complete. Furthermore, if we restrict ourselves to searching for *atomically* closed tableaux, a non-atomic formula that has been reduced need not be considered as potentially forming part of the contradiction on this branch and may be discarded.

Accordingly, we may modify our two types of rules as follows

$$\frac{(S[\alpha])}{(S\backslash\alpha, \alpha_1, \alpha_2)} \qquad \frac{(S[\beta])}{(S\backslash\beta, \beta_1) \mid (S\backslash\beta, \beta_1)}$$

where the formulas α and β must be contained in S as before but after the application α and β are deleted from the new end-points. This is denoted by the notation $S\backslash\alpha$.

In other words we eliminate signed formulas as they are reduced. Complete end-points constructed with the above rules contain signed atomic formulas only.

The connection method is based on the notion of *paths*, which we define below.

Definition 5.1: Let X be a formula, and assign $\rho(k_0) = 0$. A *path* through X is a (non-empty) set of indexed positions of X defined inductively as follows.

P0. If $X_k = X$, i.e. $k = k_0$ the root position of X,

$\{k\}$ is a path through X.

* The reader may like to visualise the positions as pointers to a *unique* concrete representation of the formula. The nodes of tableaux are then sets of pointers to formulas instead of the formulas themselves. The computational advantages of such a scheme are manifold.

If $\{S[k]\}$ is a path through X, and λk is non-atomic, then

P1. If $X_k = \neg Y$ such that $Y = X_r$,

$\{S\backslash k,r\}$ is a path through X.

P2. If $X_k = Y \vee Z$ for subformulas $Y = X_r$ and $Z = X_s$ of X,

if $\rho(k) = 0$, then $\{S\backslash k,r,s\}$ is a path through X,
if $\rho(k) = 1$, then $\{S\backslash k,r\}$ and $\{S\backslash k,s\}$ are both paths through X.

P3. If $X_k = Y \wedge Z$ for subformulas $Y = X_r$ and $Z = X_s$ of X,

if $\rho(k) = 0$, then $\{S\backslash k,r\}$ and $\{S\backslash k,s\}$ are both paths through X.
if $\rho(k) = 1$, then $\{S\backslash k,r,s\}$ is a path through X,

P4. If $X_k = Y \supset Z$ for subformulas $Y = X_r$ and $Z = X_s$ of X,

if $\rho(k) = 0$, then $\{S\backslash k,r,s\}$ is a path through X,
if $\rho(k) = 1$, then $\{S\backslash k,r\}$ and $\{S\backslash k,s\}$ are both paths through X.

If all elements of a path p are labelled by atomic formulas we say that p is an *atomic path*. In the above definition we use $S\backslash k$ to mean the set S with any occurrence of k removed. $S[k]$ indicates that k is a member of the set S.

Remark 5.2: Readers should note that Bibel's notion of path corresponds to our notion of atomic path.

Remark 5.3: In the sequel, 'a path through a formula X' we mean a path through X with respect to the initial choice of $\rho(k_0) = 0$ for the root position of X.

Under our latest modification of the tableau method the nodes of a tableau coincide exactly with paths through the formula at its root; the elements of a complete end-point constitute an atomic path. Indeed, the set of end-points on the branches of a complete (modified) tableau for X constitute all the atomic paths through X. From theorem 3.7 we get immediately a slightly modified version of Bibel's connection theorem for propositional logic.

Theorem 5.4: X is a tautology of (classical) propositional logic if and only if every atomic path through X contains a connection.

The atomic paths through the formula displayed in Fig. 1 and proved in Fig. 2 are $\{k_3, k_6, k_8\}$, $\{k_3, k_7, k_8\}$, $\{k_4, k_6, k_8\}$, $\{k_4, k_7, k_8\}$. The reader should identify the labels and polarities of these positions and note that each of these paths contain a connection.

6. CONNECTIONS VERSUS CONNECTIVES

As described above, the tableau method works by applying the reduction rules to a previously constructed tableau and checking to see if a contradiction has been derived on every branch. The nodes of a tableau are now

elements of W_x instead of signed formulas, and the reduction rules replace reduced positions by the positions resulting from their reduction. This ensures that a position (signed formula) is reduced once only.

The proof procedure is still what I shall call *connective* based, in that the decision as to which position (signed formula) is chosen for reduction next is arbitrary and not guaranteed to bring a contradiction any nearer. The specific rule applied is only dependent on the label (major connective) of the chosen position.

The final modification is to identify the contradictions *first*, and then perform those (and only those) reductions that allow the contradiction to be realised. In this way no superfluous reductions are performed. I shall call this type of proof procedure (ie. resolution-style calculi) *connection* based.

Using proposition 4.3 we can identify the potential connections (contradictions) in advance and, having done so, reduce the relevant positions in the current path so that its end-point contains the connection.

Bibel (1982a) develops a series of algorithms based on the above method whose efficiency compares favourably with most of the common proof procedures reported in the literature. Bibel's book (Bibel, 1982b) should be consulted for more details on the algorithmic aspects of the calculus.

Remark 6.1: Note that the reductions needed to realise a connection once it has been identified in the current path may be performed in any order (subject of course to the formula ordering $<_x$).

7. THE FIRST-ORDER CASE

In this section and subsequently, I shall use a (first-order) language comprising n-ary predicate constants P^n, Q^n, R^n, ...; individual variables x, y, z, ...; individual parameters a, b, c, ...; the sentential connectives as before and the *quantifiers* $\forall$ and $\exists$. Formulas are defined as usual, and formula-trees for formulas of this language are obvious generalisations of the trees introduced in §2.

The extension of the analytic tableaux method to first-order (classical) logic is based on the following four facts. Under any interpretation in a universe U,

F5. For the universal quantifier,

(a) if $\forall xY$ is false, then for some $a \in U$, Y^a_x is false,
(b) if $\forall xY$ is true, then for every $a \in U$, Y^a_x is true.

F6. For the existential quantifier,

(a) if $\exists xY$ is false, then for every $a \in U$, Y^a_x is false,
(b) if $\exists xY$ is true, then for some $a \in U$, Y^a_x is true.

Y^a_x denotes the substitution of the parameter 'a' for the free occurrences of the variable 'x' in Y.

Accordingly we get the four rules

$$\frac{(S[F\forall xY])}{(S,FY^a_x)} \text{ with proviso} \qquad \frac{(S[T\forall xY])}{(S,TY^a_x)}$$

$$\frac{(S[F\exists xY])}{(S,FY^a_x)} \qquad \frac{(S[T\exists xY])}{(S,TY^a_x)} \text{ with proviso}$$

The proviso in the F∀ and T∃ rules is that 'a' be a new parameter. Tableaux are defined as before.

In the propositional case, our first modification was to replace the signed formulas with positions. For first-order systems the situation is not as straightforward since we must take account of the parameters substituted for bound variables during the reduction of signed quantified formulas. Indeed for formulas of 'universal type' (T∀xY and F∃xY) we may need to consider different instances of Y in order to deduce the necessary contradictions*.

We follow Bibel (1980) and use an indexing mechanism to encode these instances. Each position, labelled by a quantifier of universal type, is assigned a natural number, its *multiplicity* $\mu(k)$, indicating how many instances of the sub-formula dominated by the quantifier are needed to deduce the contradictions in the tableau. Under our modification the nodes of tableaux now become *indexed* positions.

We extend our notion of polarity to positions labelled with quantifiers by adding the following two clauses to definition 4.1.

Po5. If $X_k = \forall xY$ for some subformula $Y = X_r$,

$$\rho(r) = \rho(k)$$

Po6. If $X_k = \exists xY$ for some subformula $Y = X_r$,

$$\rho(r) = \rho(k)$$

Let W^δ_x and W^γ_x denote those positions of W_x labelled by a quantifier of existential or universal type respectively. i.e.

$$W^\delta_x = \{k \in W_x \mid \lambda k = \exists(\forall) \text{ and } \rho(k) = 1(0)\}$$

$$W^\gamma_x = \{k \in W_x \mid \lambda k = \forall(\exists) \text{ and } \rho(k) = 1(0)\}$$

Definition 7.1: The *multiplicity* μ for a formula X with $\rho(k_0) = 0$, is a function which assigns a natural number $\mu(k)$ to each element $k \in W^\gamma_x$. X^μ is called an *indexed* formula and its positions W^μ_x are defined as follows $k^\varkappa \in W^\mu_x$ just in case

I1. $k \in W_x$.

I2. If $\{k_1, \ldots, k_n\} \subseteq W^\gamma_x$ are all the positions such that $k_1 < \ldots < k_n \leq k$, then $\varkappa = (j_1 \ldots j_n)$ where $1 \leq j_i \leq \mu(k_i)$, $i = 1, \ldots, n$.

The tree ordering $<_{x^\mu}$ is defined as $k^\varkappa <_{x^\mu} k'^{\varkappa'}$ just in case $k <_x k'$ and

* Recall that if ∀xY is true (∃xY is false) we may deduce that Y^a_x is true (is false) for any parameter 'a'.

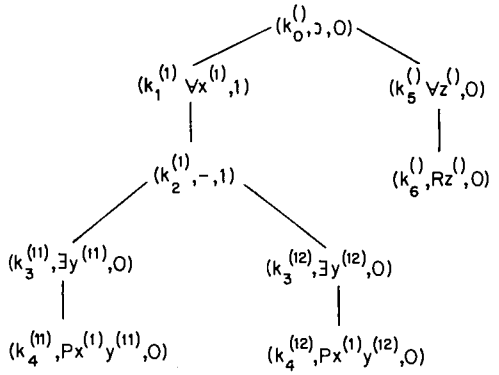


Figure 3: Indexed formula-tree for $\forall x(\neg\exists yPxy)\supset\forall zRz$

$\varkappa' = \varkappa \mathrm{conc} \tau$ for some sequence τ of natural numbers. Also we let $\lambda k^x = \lambda k$, and $\rho(k^x) = \rho(k)$.

Fig. 3 illustrates these definitions for an expansion of the formula $\forall x(\neg\exists yPxy)\supset\forall zRz$ with $\mu(k_1) = 1$ and $\mu(k_3) = 2$.

Our first two modifications now go through as before. We extend the notion of a path through a formula X to a path through an indexed formula X^μ as follows.

Definition 7.2: Let X be a formula and assign $\rho(k_0) = 0$. Let μ be a multiplicity for X. The paths through X^μ are (non-empty) sets of indexed positions of X^μ defined inductively according to the structure of X.

P0–P4. As before (definition 5.1) with indexed positions replacing the positions. If $\{S[k^x]\}$ is a path through X^μ, and λk^x is non-atomic, then

P5. If $X_k = \forall xY$ for $Y = X_r$,

if $\rho(k) = 0$, then $\{S\backslash k^x, r^x\}$ is a path through X^μ;
if $\rho(k) = 1$, then $\{S\backslash k^x, r_1^{x\cdot 1}, \ldots, r^{x\cdot\mu(k)}\}$ is a path through X^μ.

P6. If $X_k = \exists xY$ for $Y = X_r$,

if $\rho(k) = 0$, then $\{S\backslash k^x, r_1^{x\cdot 1}, \ldots, r^{x\cdot\mu(k)}\}$ is a path through X^μ;
if $\rho(k) = 1$, then $\{S\backslash k^x, r^x\}$ is a path through X^μ.

Finally, since our modified tableaux consist solely of positions, when we reduce a quantified formula we must simultaneously increment a substitution relation σ for that variable freed by the reduction. This substitution records the parameters that replace each variable in the full analytic tableau. Note that variables are distinguished by their binding position.*

8. ORDER DEPENDENCE OF RULES

The final modification, outlined in §6, was to search for the connections first, and then perform only those reductions needed to realise that connec-

* The binding position of a variable x contained in an atomic formula is the position of the quantifier $\forall x$ or $\exists x$ that immediately dominates that occurrence of x. Different instances of the same variable are distinguished by the index of the position.

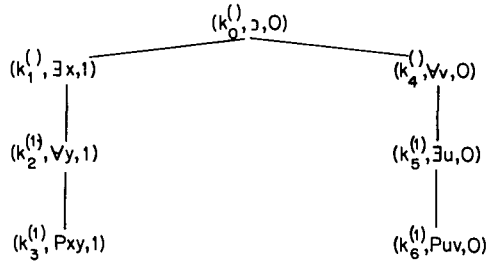


Figure 4: Representation of $\vDash x \forall y Pxy \supset \theta \forall u \exists u Puv$

tion and close the resulting branch. It was of crucial importance that the order in which the subsequent reductions are performed be immaterial.

At the first-order level this is not the case due to the restrictions placed on the existential rules. To see this consider the following example.

Example 8.1: $\vDash \exists x \forall y Pxy \supset \forall v \exists u Puv$

The positions of this formula are shown in Fig. 4 in the form of triples containing the name, label, and polarity of that position. The vertical ordering denotes the formula ordering $<_{x^\mu}$. The multiplicity μ has the value 1 throughout the formula. Here $\{k_3^{(1)}, k_6^{(1)}\}$ is the potential connection requiring that a parameter 'a' say, is substituted for both x and u; and a parameter 'b' say, for both y and v.

There are six possible reduction orders of which all but the two orders $k_1^{()}, k_2^{(1)}, k_4^{()}, k_5^{(1)}$ corresponding to (x, y, v, u), and $k_4^{()}, k_5^{(1)}, k_1^{()}, k_2^{(1)}$ corresponding to (v, u, x, y) are correct.

However, we are only interested in the *existence* of a correct reduction order. In more complicated situations, a particular order that is correct for the current set of connections may be rendered incorrect by a future connection which induces more constraining relationships between the positions. Clearly we want to avoid having to test out potential reductions each time a new connection is made.

Bibel (1982a, b) has developed a beautiful solution to this problem which we describe informally below.

Suppose a potential connection requires that the same parameter be substituted for two variables x and y. Let k_x^x and k_y^τ be the positions labelled by the binding occurrences of x and y respectively.

0. Suppose both k_x and k_y are in W_x^δ. That is to say both x and y are existentially bound. Then the connection must be rejected since no correct reduction order exists. Basically, once one position has been reduced and a parameter substituted, the reduction of the second quantifier with the same parameter must violate the proviso.
1. Suppose, w.l.o.g., that $k_x \in W_x^\gamma$ and $k_y \in W_x^\delta$; i.e. x is universally and y existentially bound. In order to satisfy the proviso when reducing k_y^τ, we must ensure that it is reduced *before* k_x^x. This situation is represented by posting the relation $k_y^\tau <^\bullet k_x^x$ i.e. k_y^τ 'before' k_x^x.
2. Now suppose both k_x and k_y are in W_x^γ; i.e. both are universally

bound. We represent the fact that the same parameter must be substituted for both variables by posting the relation $k_x^x \sim k_y^\tau$. $\sim$ is taken to be an equivalence relation, and induces more ordering constraints thus:

— if $k_z^\sigma <\bullet k_x^x$ and $k_x^x \sim k_y^\tau$ then $k_z^\sigma <\bullet k_y^\tau$

representing the fact that if k_z^σ myst be reduced before k_x^x, and the same parameter used to reduce k_y^τ as is used for k_x^x, then k_z^σ must be reduced before k_y^τ too.

Clearly the reduction order must respect $<_{x^\mu}$, the formula-tree order. Consequently we can represent all the constraints induced by connections by the reduction relation $\lhd$ defined as $\lhd = (<x^\mu \cap <\bullet)^*$.

Remark 8.2: The reader should interpret $k_x^x \lhd k_y^\tau$ to mean, k_x^x must be reduced before k_y^τ. Note that $\lhd$ is a partial order. A correct reduction sequence exists so long as $\lhd$ is *acyclic*.

Remark 8.3: If we require further that $<\bullet$ satisfy the uniqueness property if $k_z^\sigma <\bullet k_x^x$ and $k_y^\tau <\bullet k_x^x$, then $k_z^\sigma = k_y^\tau$;
we can even do without a substitution relation and explicit parameters. The set of positions k^x with $k \in W_x^\delta$ has the right uniqueness properties to function as the domain of parameters, and the extra condition above ensures that no substitution clashes occur.*

The reduction relation $\lhd$ for the example is shown in Fig. 5. The correct reduction sequences can be read off by selecting successive positions that have no unselected $\lhd$-predecessors.†

In resolution based systems, *Skolemisation* is used to encode the quantifier restrictions. In contrast to Skolemisation, the above scheme requires no normal-form. But more important is the fact that Bibel's solution to the problem represents a general method of overcoming the order dependence of reduction rules when passing from a connective based proof procedure to a connection based one. The generality of this scheme will be put to good use in the sequel.

We conclude this section with the version of the connection theorem for first-order (classical) logic that we have effectively derived.

Theorem 8.4: X is first-order valid if and only if there is a multiplicity

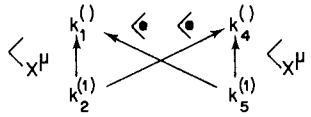


Figure 5: The reduction relations for example 8.1

* In the above development, in the interests of clarity, I have assumed that the language contains no function symbols. If functions symbols occur, $<\bullet$ cannot perform the role of a substitution. We must modify the definition of $\lhd$ such that if the term t is to be substituted for the variable x, all the binding positions of variables that occur in t are put in $<\bullet$ relation to the binding position of x.

† I.e. no arcs going out from it into (as yet) unselected positions.

μ for X, and a set of connections V in X such that

1. every atomic path through X^{μ} contains a connection from V (as a subpath), and
2. the reduction relation $\lhd$ induced by V is acyclic.

9. A CONNECTION THEOREM FOR S_5 MODAL LOGIC

A series of steps that effectively transform an analytic tableaux system for classical first-order logic into a version of Bibel's connection calculus for that logic have been described. This section contains support for the claim that the nature of these steps is not wholly dependent on the logic over which they are performed. The essential features of a connection calculus for S_5 modal logic obtained from Kanger's Gentzen system for that logic (Kanger, 1957) are presented. Lack of space necessitates that the details be kept to a minimum. A full exposition of this relatively efficient calculus will appear elsewhere.

Syntax

We augment the syntax of propositional logic with the (unary) modal operator of necessity, $\Box$.*

Semantics

Kripke's *possible world* semantics (Kripke, 1959) for S_5 yields the following facts.

M0. For necessity,

(a) if $\Box Y$ is false in a world, then Y is false in *some* world,
(b) if $\Box Y$ is true in a world, then Y is true in *every* world.

Formal system features

Kanger uses the notion of a *spotted* sequent†, where the formulas are indexed by natural numbers denoting possible worlds. For instance, the axioms of the system are sequents of the form

$$U[Y^m] \rightarrow V[Y^m]$$

and the rules for implication

$$\frac{U[Y_1^m] \rightarrow V[Y_2^m]}{U \rightarrow V,(Y_1 \supset Y_2)^m} \qquad \frac{U \rightarrow V[Y_1^m] \quad U[Y_2^m] \rightarrow V}{U,(Y_1 \supset Y_2)^m \rightarrow V}$$

* The unary modal operator, $\Diamond$ denoting possibility, can be defined as $\neg \Box \neg$.

† A sequent $U \rightarrow V$ is a notational variant of a set of signed formulas. All formulas signed T are collected in the antecedent U, those signed F are collected in the succedent V. The inference rules of a sequent system, when read from conclusion to premise(s) form a tableau system.

The rules for necessity take the form

$$\frac{U \to V[Y^m]}{U \to V,(\Box Y)^n} \text{ with proviso } \frac{U[Y^m] \to V}{U,(\Box Y)^n \to V}$$

where the proviso on the $(\to \Box)$ rule is that m must not appear in $U \to V,(\Box Y)^n$. With the rule inverted, for use as a reduction rule, the proviso reads 'm must be a *new* number' (cf. the proviso on the quantifier rules of §7.)

The reduction relation $\lhd$

The proviso on the above rule for reducing operators of possible-type* introduces an ordering constraint on the sequences of reductions in exactly the same way as the quantifiers of existential-type did before. In this case the propositional variables are 'bound' by the modal operators dominating them. Formulas bound by operators of necessary-type are true in all possible worlds and hence may appear with any index. Consequently we use a *modal multiplicity* to encode these instances.

Polarity, indexed formulas and paths are defined as before by treating the modal operators as quantifiers. Connections are pairs of positions labelled by propositional variables whose (modal) binding occurrences have been reduced with the same index. The first two steps then go through as before yielding tableaux whose nodes are sets of (unreduced) positions.

The final modification, to form a connection based proof system, requires the use of a reduction relation. Once again this relation is defined as the transitive closure of the formula-tree ordering for the indexed formula and a 'substitution' ordering $<\bullet$ induced by the connections. The domain of positions labelled by modal operators of possible-type replaces the natural numbers of Kanger's system as names for the possible worlds in the same way as the indexed elements of W^{δ}_{x} could be used to replace the domain of parameters before.†

We conclude with a statement of the connection theorem for S_5 modal logic obtained from Kanger's system.

Theorem 9.1: X is S_5 valid if and only if there is a modal multiplicity ν for X, and a set of connections V in X^{ν} such that

1. every atomic path through X^{ν} contains a connection from V (as a subpath), and
2. the S_5 reduction relation $\lhd$ induced by V is acyclic.

10. CONCLUSIONS AND FURTHER WORK

I have presented a detailed account of the relationship between a system of

* Occurrences of $\Box$ (resp. $\Diamond$) with polarity 0(resp. 1).

† This tight correspondence between the structures supporting a connection calculus for S_5 and those for first-order classical logic is not accidental; there is a fundamental relationship between S_5 and the monadic predicate calculus aptly illuminated by the present analysis.

analytic tableaux and a connection calculus for first-order classical logic. The relationship was developed by applying a series of modifications to the former to obtain the latter. I have claimed that these modifications are general enough to apply to other logics specified using tableau or Gentzen systems and, to substantiate this, preliminary results in the form of a connection calculus for S_5 modal logic have been presented briefly.

This work is being extended in two complementary directions. Firstly, the theoretical foundation of the methodology itself is being investigated in the hope of characterising precisely the conditions under which a formal system may be systematically automated in the above manner. Secondly the methodology is being refined by applying it to other common (first-order) logics such as S_4 modal logic and first-order intuitionistic logic.

REFERENCES

Andrews, P. B. (1981). Theorem-Proving via General Matings, *J of the ACM*, **28** (2), 193–214.

Beth, E. W. (1959). *The foundations of mathematics*, North Holland, Amsterdam. Also Harper and Row, New York 1966.

Bibel, W. (1980). The Complete Theoretical Basis for the Systematic Proof Method. Bericht ATP-6-XII-80. Tchnische Universität München.

Bibel, W. (1981). On Matrices with Connections, *J. of the ACM*, **28**(4), 633–645.

Bibel, W. (1982a). A Comparative Study of Several Proof Procedures, *Artificial Intelligence*, **18**, 269–293.

Bibel, W. (1982b). *Automated Theorem Proving*. Friedr Vieweg & Sohn, Braunschweig/Wiesbaden.

Boyer, R. S., and Moore, J. S. (1972). The Sharing of Structure in Theorem-Proving Programs. *Machine Intelligence*, **7**, 101–116.

Fitting, M. C. (1983). Proof methods for modal and intuitionistic logics, *Synthese library*, **169**: D Reidel, Dordrecht, Holland.

Gentzen, G. (1969). Investigations into Logical Deduction. In *The collected papers of Gerhard Gentzen*, chapter 3, pp 68–131. Szabo M E (ed), North Holland, Amsterdam, Translation.

Kanger, S. (1957). Provability in logic, *Stockholm Studies in Philosophy*, **1**: Almqvist and Wiksell, Stockholm.

Kripke, S. (1959). A Completeness Theorem in Modal Logic, *J Symbolic Logic*, **24**(1), 1–14;

Nishimura, H. (1983). Hauptsatz for higher-order modal logic. *J Symbolic Logic*, **48**(3), 744–751.

Robinson, J. A. (1965). A machine oriented logic based on the Resolution principle, *J of the ACM* **12**, 23–41.

Smullyan, R. M. (1968). First-Order Logic. *Ergebnisse der Mathematik*, **43**, Springer-Verlag, Berlin.

Smullyan, R. M. (1970). *Abstract quantification theory. Intuitionism and Proof Theory*, North Holland, Amsterdam, pp **79–91**.

FORMULATING PROOF SYSTEMS FOR AUTOMATED DEDUCTION[1]

L. A. Wallen[2]

We describe various logic-independent techniques for the design of proof systems that support efficient automated proof search. The techniques are illustrated through the design of a proof system for a fragment of an intensional logic.

1. Introduction

Symbolic logic plays a major rôle in computer science, both as a tool for analysis and as a framework for representation and reasoning. A pragmatic view is adopted when logic is used as a representation language: there is not one logic, but many; a logic is a very general theory that axiomatises the basic properties of the objects of semantic interest. Such applications of logics abound: *eg.,* classical and constructive logic for program derivation [Mar82,Abr84], modal logics for reasoning about distributed processes [HM84], higher-order logic for hardware verification [Gor85], *etc.*

It is the proof system for a logic that furnishes us with the concrete notion of "proof." Within the bounds of soundness, we can design the proof system to reflect our concern with pragmatic issues such as elegance, or – and this is our present concern – efficiency of proof search.

In many applications the computer can be used to support the search for proofs. In some circumstances it is appropriate to fully automate search. Within interactive proof environments, for example, automation of the search for certain types of subproof can be used to raise the level of interaction, leaving the user free to concentrate on more demanding aspects of proof design (see *eg.,* [BM79,GMW79,Con86].)

Here we meet with a problem. Proof systems that directly reflect the underlying semantics of logics, such as sequent calculi are easy to design because they model the semantics of a logic more or less directly. As a consequence, the majority of logics considered for application in computer science admit such proof systems. However, sequent systems (and systems of natural deduction in general) form inadequate bases for automated proof search as we shall show below. Moreover, while relatively efficient proof systems exist for classical logic (*eg.,* resolution), the extension of such methods to other logics has met with limited success [Far86].

In this short paper we outline elements of a theory of proof system design which can be used to develop efficient proof systems for a given logic directly from a sequent proof system for that logic. We are thus able to design an efficient proof system for a logic by sole virtue of the fact that it admits a certain type of sequent proof system. In this way we are able to treat logics that were previously outside the scope of traditional (automated) theorem-proving techniques. We illustrate the theory through the design of a proof system for a fragment of an intensional logic.

2. A simple intensional logic

In this section we introduce a simple intensional logic which we call **L**. The language of **L** consists of denumerably many propositional variables $p, q, r, \ldots$, a binary connective $\supset$ and a unary connective $\Box$. The set of *lower* formulae of **L** is the smallest set containing the propositional variables and closed under the usual formation rule for the binary connective $\supset$. The set of formulae of **L** is the smallest set containing the lower formulae, closed under both the formation rule for $\supset$ and the rule: if A is a *lower* formula, $\Box A$ is a formula. Basically, we exclude nested modalities. The propositional variables will sometimes be called *atomic* formulae. We use A, B, C and Γ, Δ as metavariables to range over formulae and sets of formulae respectively.

We give a Kripke-style semantics to **L**. An **L**-frame is a pair $\langle G, R\rangle$ consisting of a non-empty set G and a reflexive relation R over G. An **L**-model is a triple $\langle G, R, \models\rangle$ where $\langle G, R\rangle$ is an **L**-frame and $\models$ a relation between elements of G and formulae such that: for all $w \in G$

1. Either $w \models A$ or $w \not\models A$, for all atomic A.
2. $w \models A \supset B$ iff $w \models B$ or $w \not\models A$.
3. $w \models \Box A$ iff for all $v \in G$ such that $w\ R\ v$, $v \models A$.

[1] *IEE Colloquium on Theorem Provers in Theory and Practice*, IEE, Savoy Place, London, March 1987.

[2] Lincoln Wallen is a member of the Department of Artificial Intelligence, University of Edinburgh.

A model $\langle G, R, \models \rangle$ satisfies a formula A just when $w \models A$, for all $w \in G$. A formula A is an **L**-consequence of a set of formulae Γ just in case, every **L**-model that simultaneously satisfies the formulae of Γ also satisfies A.

We can define a natural proof system for the **L**-consequence relation based on *sequents*. If Γ and Δ are sets of formulae of **L**, then $\Gamma \rightarrow \Delta$ is a sequent. Γ is the *antecedent* and Δ the *succedent*. We interpret sequents as follows:

4. $w \models \Gamma \rightarrow \Delta$ iff if $w \models \Gamma$ (*i.e.*, $w \models A$ for every element $A \in \Gamma$), then for some A in Δ, $w \models A$.

To decide whether a formula A is an **L**-consequence of a set of formulae Γ, we attempt to prove the sequent $\Gamma \rightarrow A$ using the proof system defined below. We take as axioms all sequents of the form:

$$\Gamma, A \rightarrow A, \Delta$$

for atomic A; and as operational rules:

$$\frac{\Gamma, A \rightarrow B, \Delta}{\Gamma \rightarrow A \supset B, \Delta} \rightarrow \supset \qquad \frac{\Gamma, B \rightarrow \Delta \quad \Gamma \rightarrow A, \Delta}{\Gamma, A \supset B \rightarrow \Delta} \supset \rightarrow$$

$$\frac{\Gamma_{\Box} \rightarrow A}{\Gamma \rightarrow \Box A, \Delta} \rightarrow \Box \qquad \frac{\Gamma, A \rightarrow \Delta}{\Gamma, \Box A \rightarrow \Delta} \Box \rightarrow$$

where $\Gamma_{\Box} =_{df} \{B \mid \Box B \in \Gamma\}$. Derivations are defined as usual and a proof is a derivation whose leaves are axioms. The root of a derivation is called the *end sequent*. In practice we use the rules inverted, working backwards from the desired end sequent to the leaves. The application of a rule, inverted in this manner, will be called a *reduction*.

It is easy to show that the above proof system is a characterisation of the consequence relation of **L**.

3. Rule order and unification.

Consider the following pair of derivations, where we have "boxed" the principal formula of each reduction (recall that derivations are being constructed from their root to their leaves):

$$\dfrac{\dfrac{\dfrac{\times}{\boxed{q} \rightarrow \boxed{q}}}{\boxed{p \supset q} \rightarrow q}}{p, \Box(p \supset q) \rightarrow \boxed{\Box q}} \qquad \dfrac{\dfrac{\times \qquad \times}{p, \boxed{q} \rightarrow \boxed{q} \qquad \boxed{p} \rightarrow \boxed{p}, q}}{p, \boxed{p \supset q} \rightarrow q}$$

$$\boxed{\Box p}, \Box(p \supset q) \rightarrow \Box q \qquad\qquad \Box p, \Box(p \supset q) \rightarrow \boxed{\Box q}$$

We cannot obtain a proof from the derivation on the left because we are unable to close one of its leaves. This problem arises because the application of the $\rightarrow \Box$ rule restricted the formulae available for such a completion. We can influence the content of the set $\Gamma_{\Box}$ by changing the order of rule application so that more (or less) formulae of the form $\Box A$ occur in the antecedent at the application of the $\rightarrow \Box$ rule. This we have done in the proof on the right. The import for automated proof search is that we must consider all possible permutations of rule applications as potentially leading to a proof.

We can overcome this type of problem in the following manner. An *indexed* formula is a pair A^{α} comprising a formula A and an element α of a set formed from the disjoint union of two alphabets $\mathcal{C}$ and $\mathcal{V}$. Let Γ and Δ range over sets of indexed formulae and α, β over indices. The basic idea is to index formulae rather than remove them from a sequent, construct a "proof" with no regard for rule order, then employ unification to ensure that an appropriate rule order could have been chosen that would result in a proof in the original sequent system. The operational rules of the new proof system are as follows:

$$\frac{\Gamma, A^{\alpha} \rightarrow B^{\alpha}, \Delta}{\Gamma \rightarrow (A \supset B)^{\alpha}, \Delta} \rightarrow \supset \qquad \frac{\Gamma, B^{\alpha} \rightarrow \Delta \quad \Gamma \rightarrow A^{\alpha}, \Delta}{\Gamma, (A \supset B)^{\alpha} \rightarrow \Delta} \supset \rightarrow$$

$$\frac{\Gamma \rightarrow A^{b}, \Delta}{\Gamma \rightarrow (\Box A)^{\alpha}, \Delta} \rightarrow \Box \qquad \frac{\Gamma, A^{x} \rightarrow \Delta}{\Gamma, (\Box A)^{\alpha} \rightarrow \Delta} \Box \rightarrow$$

where we require that the elements $a \in \mathcal{C}$ and $x \in \mathcal{V}$ used to extend indices in the $\Box$ rules do not appear as indices of any other formula in the upper sequent. The axiom rule is:

$$\frac{\sigma}{\Gamma, A^{\alpha} \rightarrow A^{\beta}, \Delta}$$

for atomic A, where $\sigma: \mathcal{V} \mapsto \mathcal{C}$ is such that $\sigma(\alpha) = \sigma(\beta)$. Derivations are defined as usual. Proofs, on the other hand, are derivations whose leaves are not only instances of the axiom rule, but for which the mappings associated with the set of leaves are consistent. To prove that A is an L-consequence of Γ we prove the sequent $\Gamma^a \rightarrow A^a$ for some arbitrary element $a \in \mathcal{C}$. ($\Gamma^a =_{df} \{A^a \mid A \in \Gamma\}$.)

A proof of our example is shown below. Notice how we have used the previously incorrect order for applying rules.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\{y \rightarrow b\}}{p^x, \boxed{q^y} \rightarrow \boxed{q^b}} \quad \dfrac{\{x \rightarrow b, y \rightarrow b\}}{\boxed{p^x} \rightarrow \boxed{p^y}, q^b}}{p^x, \boxed{(p \supset q)^y} \rightarrow q^b}}{p^x, \boxed{(\Box(p \supset q))^a} \rightarrow q^b}}{p^x, (\Box(p \supset q))^a \rightarrow \boxed{(\Box q)^a}}}{\boxed{(\Box p)^a}, (\Box(p \supset q))^a \rightarrow (\Box q)^a}$$

We can use a simple unification algorithm to compute the appropriate mappings when we reach the leaves of derivations and a substitution table to check consistency.

4. Representational redundancy

We have successfully reduced the space to be searched for a proof by identifying derivations that differ only in the order in which the modal rules are applied. We shall remove remaining redundancies of a similar nature in the next section. In this section we deal with an issue of representation.

First note that both the sequent systems we have discussed possess the so-called *subformula* property. That is, derivations are formed solely from subformulae of the end sequent. Consequently, every formula occurrence in a derivation has a unique *image* in the end sequent.

In our search for a proof we need to represent trees of sequents which occupy large amounts of computer memory. We can improve this situation by storing the end sequent explicitly in memory once, but instead of storing new sequents generated by rule applications explicitly, we construct them out of pointers to subformulae in the end sequent. This is a principled application of *structure-sharing* [BM72]: a technique developed within the resolution community to cope with a similar problem. Here we see it arise naturally from proof-theoretical properties of our sequent system.

5. Primary and incidental choices

Although the new sequent system we have introduced successfully identifies derivations that differ only in the order in which the modal rules are applied, and with the use of structure-sharing we can reduce the memory requirements for individual derivations, an implementation of this system would still suffer from its emphasis on connectives. That is, although at all times we are working towards instances of the axiom schema the proof system does not reflect this. We could spend time and space reducing a formula in a sequent from which there is no hope of obtaining an axiom given the other formulae in that sequent.

A derivation of an end sequent is said to be *complete* just when its leaves consist solely of atomic formulae; *i.e.,* no more reductions are possible. The index of an atomic formula appearing in a derivation is chosen at the reduction of the modal operator that dominates its' image in the end sequent (if there is no such operator the index of the atomic formula is the index chosen at the start of the derivation for the formulae of the end sequent). If we ensure that two occurrences of modal operators that share the same image are reduced using the same index, then every complete derivation of that end sequent has the *same* set of leaves. Since the criterion for distinguishing a proof from a derivation is that all the leaves are axioms (under some mapping of indices) we have only to check whether every leaf in this set is an axiom. In other words, the order in which we apply the sequent rules is immaterial! The rules are merely a manner of enumerating this unique set of leaves which we shall call *atomic paths.*

Based on these observations we can devise more efficient means of checking whether the atomic paths are axioms (under a given mapping of indices).

First, we define positive and negative occurrences of formulae as follows: C occurs positively in C; C occurs positively (negatively) in $A \supset B$ if it occurs negatively (positively) in A or positively (negatively) in B; C occurs positively (negatively) in $\Box A$ if it occurs positively (negatively) in A; C occurs positively (negatively) in $\Gamma \rightarrow \Delta$ if it occurs positively (negatively) in A for some $A \in \Gamma$ or negatively (positively) in A for some $A \in \Delta$.

Now notice that a formula occurs in an antecedent (succedent) within a derivation if and only if its image occurs positively (negatively) in the end sequent. This means that we can determine in advance whether a given

atomic subformula of the end sequent is constrained to occur in antecedents or succedents within derivations. A *connection* is a pair of atomic formula occurrences $\{A_+^\alpha, A_-^\beta\}$ in an atomic path, where we have used subscripts to indicate whether the images of these formulae occur positively or negatively in the end sequent. Such a connection is said to be *complementary* just in case there is a mapping σ such that $\sigma(\alpha) = \sigma(\beta)$. If a connection is complementary then, the paths that contain it are axioms. Consequently we have:

> A formula A is an **L**-consequence of a set of formulae Γ if there is a mapping and a set of connections simultaneously complementary under that mapping, such that every atomic path for $\Gamma \rightarrow A$ contains a connection from the set.

We can enumerate the atomic paths for a given end sequent by viewing the latter as a nested *matrix*. If $A \supset B$ occurs positively (negatively) in the end sequent, we place the matrix representations of A and B in a 1×2 (2×1) matrix. For atomic formulae we indicate whether it occurs positively or negatively in the end sequent by means of a subscript. We also indicate the indices for atomic formulae which can be determined from their dominating modal operator. As a matrix therefore, our example reads:

$$\Box p_+^x \qquad \Box \begin{pmatrix} p_-^y \\ q_+^y \end{pmatrix} \qquad \Box q_-^b$$

Notice how the atomic formulae p_-^y and q_+^y of the middle column have the same index y. This is because they share the same dominating modal operator which occurs positively in the end sequent and hence will appear only in antecedents within derivations.

The atomic paths for an end sequent are the horizontal matrix paths through its matrix representation. Our example matrix has a total of two atomic paths: $\{p_+^x, p_-^y, q_-^b\}$ and $\{p_+^x, q_+^y, q_-^b\}$.

Our proof system has now changed drastically. No longer are we interested in connectives. To prove our example consequence we choose an atomic formula in the matrix, say p_+^x, and then look for another occurrence of the propositional variable p occurring negatively. In this case there is only one, namely p_-^y. We then attempt to unify the indices obtaining the substitution x/y. All paths containing the connection $\{p_+^x, p_-^y\}$ are eliminated from consideration. Next we choose an atomic formula not on such a path, say q_+^y, and repeat the process resulting in the addition of the connection $\{q_+^y, q_-^b\}$ to our set and the component b/y to our current substitution. Since we have covered all atomic paths with our two connections and can construct a consistent mapping (from the consistent substitution), we conclude that $\Box q$ is an **L**-consequence of $\Box p$ and $\Box(p \supset q)$.

6. Conclusions

We have outlined some techniques that may be used to develop an efficient proof system for a given logic from a sequent proof system for that logic. These techniques have been applied successfully to develop efficient proof systems for a range of widely used first-order intensional logics [Wal87]. Implementation of one of these systems has been completed and tested [WW87]. This represents the most comprehensive application of theorem-proving techniques to non-standard logics yet achieved.

Two further points are worth mentioning. Firstly, the resulting proof systems require no normal-forming of input. Secondly, due to their genesis from sequent systems it is possible to interpret the path-checking algorithms (*i.e.*, proof search) as constructing derivations in indexed sequent systems similar to the one introduced above. This facilitates the use of the matrix systems within interactive environments.

These techniques have their origin in the matrix methods of Andrews [And81] and Bibel [Bib81] for classical logics. In [Bib82], Bibel develops many *path-checking* algorithms for classical logic and relates them favourably to standard resolution systems. All of these algorithms may be used in the matrix proof systems for non-standard logics developed along the lines outlined above. Fitting [Fit83] develops tableau systems for various modal logics. His systems do not incorporate unification and are connective based. These tableau systems therefore suffer from the same redundancies as the sequent systems we started with.

Acknowledgements

This work was supported in part by SERC/Alvey grants GR/D/44874 and GR/D/44270.

References

[Abr84] J.R. Abrial. The mathematical construction of a program. *Science of Computer Programming*, 4:45–86, 1984.

[And81] P.B. Andrews. Theorem-proving via general matings. *Journal of the Association for Computing Machinery*, 28(2):193–214, April 1981.

[Bib81] W. Bibel. On matrices with connections. *Journal of the Association for Computing Machinery*, 28(4):633–645, October 1981.

[Bib82] W. Bibel. A comparative study of several proof procedures. *Artificial Intelligence*, 18:269–293, 1982.

[BM72] R.S. Boyer and J.S. Moore. The sharing of structure in theorem-proving programs. In B. Meltzer and D. Michie, editors, *Machine Intelligence 7*, pages 101–116, Edinburgh University Press, 1972.

[BM79] R.S. Boyer and J.S. Moore. *A Computational Logic. ACM monograph series*, Academic Press, 1979.

[Con86] R.L. Constable *et al. Implementing Mathematics with the Nuprl Proof Development System.* Prentice Hall, 1986.

[Far86] L. Fariñas-del-Cerro. Resolution modal logics. *Logique et Analyse*, 110-111:153–172, 1986.

[Fit83] M.C. Fitting. *Proof methods for modal and intuitionistic logics.* Volume 169 of *Synthese library*, D. Reidel, Dordrecht, Holland, 1983.

[GMW79] M.J.C. Gordon, A.J. Milner, and C.P. Wadsworth. *Edinburgh LCF – A mechanised logic of computation.* Volume 78 of *Lecture Notes in Computer Science*, Springer Verlag, 1979.

[Gor85] M.J.C. Gordon. *HOL: A machine oriented formulation of higher-order logic.* Report 68, Computer Lab., Cambridge University, 1985.

[HM84] J.Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. In *3rd ACM Conference on the Principles of Distributed Computing*, pages 50–61, 1984.

[Mar82] P. Martin-Löf. *Constructive mathematics and computer programming*, pages 153–175. Volume IV of *Logic, Methodology and Philosophy of Science*, North-Holland, Amsterdam, 1982.

[Wal87] L.A. Wallen. Matrix proof methods for modal logics. In J. McDermott, editor, *10th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann Inc., 1987. To appear.

[WW87] L.A. Wallen and G.V. Wilson. A computationally efficient proof system for S5 modal logic. In J. Hallam and C. Mellish, editors, *Advances in Artificial Intelligence*, John Wiley & Sons, 1987. Proceedings of AISB87, Edinburgh, Scotland, April 1987, to appear.

A Computationally Efficient Proof System for S5 Modal Logic

Lincoln A. Wallen Gregory V. Wilson
Department of Artificial Intelligence
Edinburgh University
Scotland

Abstract

We present a computationally efficient matrix proof system for S5 modal logic. The system requires no normal-form and admits a natural implementation using structure-sharing techniques. In addition, proof search may be interpreted as constructing generalised proofs in an appropriate sequent calculus, thus facilitating its use within interactive environments. We describe features of an implementation developed from an existing implementation of a matrix proof system for first-order logic.

1 Introduction.

Modal logics are widely used in various branches of artificial intelligence and computer science as logics of knowledge and belief (*eg.*, [Moo80,HM85,Kon84]), logics of programs (*eg.*, [Pne77]), and for specifying distributed and concurrent systems (*eg.*, [HM84,Sti85]). As a consequence, the need arises for proof systems for these logics which facilitate efficient automated proof search.

The main hurdle to the application of resolution based techniques to non-standard logics is that the techniques are formulated under the assumption that the input formulae are in clausal form [CL73]. Most non-standard logics of interest fail to admit such a normal-form.

Bibel's connection calculus [Bib81,Bib82a] is a non-clausal proof system for first-order logic comparable in computational efficiency to the most efficient of the clausal techniques for that logic [Bib82b]. In [Wal86] it was shown that far from being an ad-hoc proof system for one particular logic, the connection calculus could be seen as a framework for implementing sequent- and tableau-based proof systems in a computationally efficient manner. Since most of the non-standard logics of interest admit such proof systems we can make use of this analysis to develop connection calculi for them. This paper fulfills an undertaking to present the details of such an application to S5 modal logic.

We begin by presenting Kanger's sequent calculus for S5 [Kan57] using a notation developed by Smullyan and Fitting [Smu68,Fit72]. Next, we develop a connection calculus from the sequent system using the techniques discussed in [Wal86]. In addition to the basic theory, we give details of an implementation developed directly from an existing implementation of the connection calculus for standard first-order logic. The adaptation from first-order classical logic to S5 proved quite straightforward. Further details of this implementation can be found in [Wil86].

Because we do not need to give up our sequent interpretation of the proof system, it is possible to view proof search within the modal connection calculus as a process which constructs a form of sequent proof tree. Brief details of this are given. Such a facility encourages the use of these techniques within interactive environments (*cf.* [BT75]).

Advances in Artificial Intelligence, editors J. Hallam and C. Mellish, pages 141–153, John Wiley & Sons, 1987. Proceedings of AISB87, Edinburgh Scotland, April 1987.

2 Preliminaries.

2.1 Syntax, semantics and notation.

Modal *formulae* are defined as usual by adding the formation rule:

if A is a formula, then so are $\Box A$ and $\Diamond A$,

to the formation rules for propositional formulae. We let A, B, C range over modal formulae.

A *signed* modal formula is a pair $\langle A, n\rangle$, where A is a formula and $n \in \{0, 1\}$. We let X, Y, Z range over signed modal formulae.

Following Smullyan [Smu68] and Fitting [Fit83] we classify signed modal formulae and their principal signed subformulae as follows:

α	α_1	α_2
$\langle A \wedge B, 1\rangle$	$\langle A, 1\rangle$	$\langle B, 1\rangle$
$\langle A \vee B, 0\rangle$	$\langle A, 0\rangle$	$\langle B, 0\rangle$
$\langle A \Rightarrow B, 0\rangle$	$\langle A, 1\rangle$	$\langle B, 0\rangle$
$\langle \neg A, 1\rangle$	$\langle A, 1\rangle$	$\langle A, 1\rangle$
$\langle \neg A, 0\rangle$	$\langle A, 0\rangle$	$\langle A, 0\rangle$

β	β_1	β_2
$\langle A \wedge B, 0\rangle$	$\langle A, 0\rangle$	$\langle B, 0\rangle$
$\langle A \vee B, 1\rangle$	$\langle A, 1\rangle$	$\langle B, 1\rangle$
$\langle A \Rightarrow B, 1\rangle$	$\langle A, 0\rangle$	$\langle B, 1\rangle$

ν	ν_0
$\langle \Box A, 1\rangle$	$\langle A, 1\rangle$
$\langle \Diamond A, 0\rangle$	$\langle A, 0\rangle$

π	π_0
$\langle \Box A, 0\rangle$	$\langle A, 0\rangle$
$\langle \Diamond A, 1\rangle$	$\langle A, 1\rangle$

We shall use $\alpha, \alpha_1, \alpha_2, \beta, \beta_1, \beta_2, \ldots$ to denote signed formulae and their components of the respective types.

For example, a signed formula of the form $\langle A \wedge B, 1\rangle$ is an α according to the first table above. Its components $\langle A, 1\rangle$ and $\langle B, 1\rangle$ are denoted by α_1 and α_2 respectively.

An S5-*model structure* is a pair $\langle G, R\rangle$ consisting of a set G and an equivalence relation R on G. An S5-model is a triple $\langle G, R, \Vdash\rangle$ where $\langle G, R\rangle$ is an S5-model structure, and $\Vdash$ is a relation between elements of G and signed formulae which, for all $w \in G$ satisfies:

1. exactly one of $w \Vdash \langle A, 1\rangle$ or $w \Vdash \langle A, 0\rangle$;
2. $w \Vdash \alpha$ iff $w \Vdash \alpha_1$ and $w \Vdash \alpha_2$;
3. $w \Vdash \beta$ iff $w \Vdash \beta_1$ or $w \Vdash \beta_2$;
4. $w \Vdash \nu$ iff for all $v \in G$ such that wRv, $v \Vdash \nu_0$;
5. $w \Vdash \pi$ iff for some $v \in G$ such that wRv, $v \Vdash \pi_0$.

A formula A is S5-*valid* in the S5-model $\langle G, R, \Vdash\rangle$ iff for all $w \in G$, $w \Vdash \langle A, 1\rangle$. A formula is S5-*valid* iff it is S5-valid in all S5-models.

2.2 A sequent calculus for S5.

Let $\langle G_0, R_0\rangle$ be an S5-model structure, fixed for the rest of this section. We use p, q to denote elements of G_0. We refer to these elements as *prefixes*. For $p \in G_0$, pX is called a *prefixed* signed formula. Prefixes are used to name possible worlds in some arbitrary model. R_0 is used to represent the relation of accessibility between prefixes and hence between possible worlds in that model. pX is satisfied by a model just when the world denoted by p, say $\iota(p)$, satisfies X; *i.e.*, $\iota(p) \Vdash X$.

For our purposes *sequents* are merely sets of prefixed signed formulae. We use S, pX to denote $S \cup \{ pX \}$. Note that pX may occur in S.

We now present Kanger's system for S5 using this notation. The basic sequent is a sequent which contains both $p\langle A, 1\rangle$ and $p\langle A, 0\rangle$ for some atomic formula A, *i.e.*,

$$S, p\langle A, 1\rangle, p\langle A, 0\rangle.$$

The operational rules are stated concisely thus:

$$\frac{S,\ p\alpha_1, p\alpha_2}{S,\ p\alpha}\ \alpha \qquad \frac{S,\ p\beta_1 \quad S,\ p\beta_2}{S,\ p\beta}\ \beta$$

$$\frac{S,\ q\nu_0}{S,\ p\nu}\ \nu \qquad \frac{S,\ q\pi_0}{S,\ p\pi}\ \pi$$

The π-rule is subject to the following proviso:

> q must not occur in S (*i.e.*, q must not prefix any other formula in S).

Derivations and *proofs* are defined as usual. A formula A is a *theorem* if there is a proof of the sequent $\{\, p\langle A, 0\rangle \,\}$ for some (arbitrary) prefix p. This calculus is both sound and complete for S5 [Kan57].

Remark. The usual representation of sequents can be recovered by introducing a *sequent arrow* $\longrightarrow$ and placing those prefixed formulae signed 1 on the left of the arrow and those signed 0 on the right.

In practice we use the rules from conclusion to premises; *i.e.*, we start with the sequent $\{\, p\langle A, 0\rangle \,\}$ and build a proof tree backwards from this root to the leaves. Used in this way, the sequent system is equivalent to Fitting's prefixed tableau system for S5 [Fit72]. The force of the proviso on the π-rule is that the prefix q must be completely new to the upper sequent. ■

3 A connection calculus for S5.

As discussed in [Wal86] the proviso on the π rule introduces an order dependence in the search for a proof. If a prefix is introduced by the use of the ν rule it cannot subsequently be introduced by the use of the π rule. The path to an basic sequent may therefore be blocked by injudicious choices of the prefix q in both (inverted) modal rules. This fact, together with the blind *connective-driven* search for the appropriate pairs of atoms to form a basic sequent, makes direct implementation of the sequent system inefficient. In [Wal86] we show how inefficiencies such as these are tackled in the case of first-order logic by Bibel's connection calculus.

In the context of modal logic the key techniques are:

- *connection driven search:* putative instances of the basic sequent are identified using structural properties of the formula.
- *delayed choice of prefix:* the symbols introduced as prefixes during use of the ν rule are considered as "variables." The prefixes introduced by applications of the π rule are considered as "constants." The instantiation of the variable prefixes is driven by the choice of connection (basic sequent) which requires the prefixes of the distinguished atomic formulae to be identical.
- *reduction ordering:* the substitution mentioned above is considered admissible if a correct (partial) proof tree can always be constructed from the current set of connections. In particular this means respecting the proviso on the π-rule. This property of the substitution is checked directly rather than by actually constructing such a proof tree.
- *structure sharing:* the intermediate states of the proof are encoded using pointers into the original formula. No new formulae need be considered. In particular, multiple instances of subformulae (in this case instances of formulae dominated by modal operators) are obtained by an indexing method rather than explicit copying.

3.1 Formula trees.

A *formula tree* for a signed modal formula is a variant of its formation tree containing additional information as to the polarity of its subformulae. It is best explained by example. **Figure 1** shows the formula tree for the formula $\langle \Box((A \wedge B) \wedge \Diamond C \Rightarrow \Diamond(A \wedge C)), 0\rangle$. The piece of

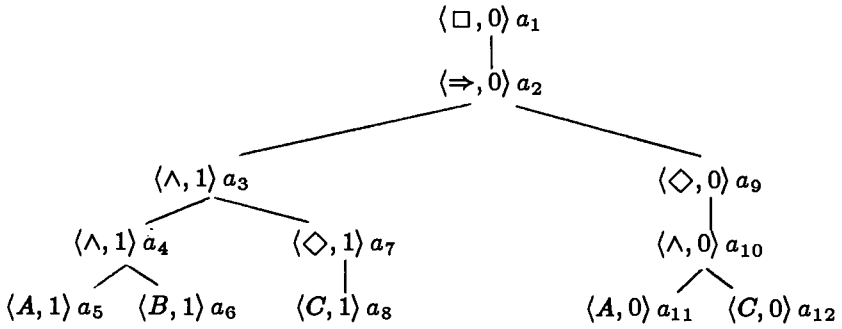


Figure 1: A formula tree.

concrete syntax associated with a node or *position* in the tree is called the *label* of that position.

We classify the positions of a formula tree according to the classification of the subformula rooted at that position. Thus if the subformula is an α, then the position is an α; if it is a β, the position is a β, *etc.* Additionally, we classify the root node of the formula tree as a π_0.

Notice that each position has two types: its principal type (*eg.*, $\alpha, \beta, \nu, \ldots$) is determined by its label and polarity, while its secondary type (*eg.*, $\alpha_1, \alpha_2, \beta_1, \ldots$) arises from the type of its parent.

For a given formula tree we use k, l, possibly subscripted, to denote positions and $\mathcal{V}_0$ and Π_0 denote the sets of positions of type ν_0 and π_0 respectively.

3.2 Modal multiplicity.

The inverted rule enables any prefix q to be introduced into a sequent to prefix formulae rooted at ν_0-type positions. Within the context of a sequent proof this is the mechanism by which one works towards instances of the basic sequent to complete a branch.

The definitions of this section are introduced for a given formula tree for a given signed formula X. A function μ_M from $\mathcal{V}_0$ to the positive integers is called a *modal multiplicity* for X. A modal multiplicity serves to encode the number of distinct instances of ν_0-type (sub)formulae used within a putative proof.

If μ_M is a modal multiplicity for X we define the (indexed) formula tree for the *indexed formula* X^{μ_M} as a tree of indexed positions of the form k^κ, where k is a position of the basic formula tree for X and κ is a sequence of positive integers defined as follows: if $k_1 < k_2 < \cdots < k_n \leq k$, $0 \leq n$, are those ν_0-type positions that dominate k in the basic formula tree for X, then

$$\kappa \in \{ (j_1\, j_2 \cdots j_n) \mid 1 \leq j_i \leq \mu(k_i),\ 1 \leq i \leq n \}.$$

The ordering in the indexed tree $<^{\mu_M}$ is defined in terms of the ordering on the underlying tree. For indexed positions k^κ and l^τ

$$k^\kappa <^{\mu_M} l^\tau \quad \text{iff} \quad k < l \quad \text{and} \quad \tau = \kappa\theta,$$

where θ is some sequence of positive integers. The polarity and label of an indexed position k^κ is taken to be the same as the polarity and label of its underlying position. Consequently, indexed positions inherit the type of their underlying position.

We let u, v, possibly subscripted, range over indexed positions when we are not interested in the index, and omit the superscript on $<$. We abuse our notation and let $\mathcal{V}_0$, Π_0, *etc.*, denote the sets of indexed positions of an indexed formula tree of the appropriate types. Henceforth we shall refer to indexed positions simply as positions.

Figure 2 shows the indexed formula tree for our example formula with a modal multiplicity of $\mu_M(a_{10}) = 2$ and 1 otherwise. As a convention we omit empty indices.

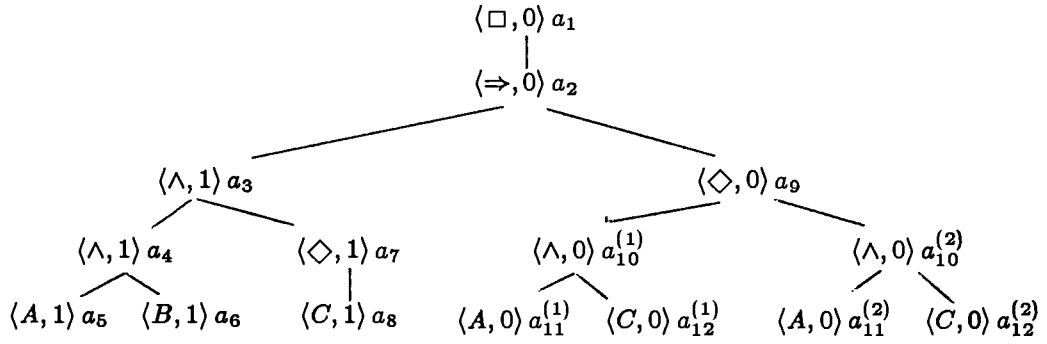


Figure 2: An indexed formula tree.

3.3 Paths and connections.

A *path through* X^{μ_M} is a subset of the positions of its formula tree. We shall use s, t, possibly subscripted, to denote these paths, and adopt the notation $s[u]$ for a path s with an occurrence of the position u. The set of paths through X^{μ_M}, is the smallest set such that:

1. $\left\{ k_0^{()} \right\}$ is a path, where $k_0^{()}$ is the root node of the formula tree for X^{μ_M};
2. if $s[\alpha^\kappa]$ is a path, so is $(s - \{\alpha^\kappa\}) \cup \{\alpha_1{}^\kappa, \alpha_2{}^\kappa\}$;
3. if $s[\beta^\kappa]$ is a path, so are $(s - \{\beta^\kappa\}) \cup \{\beta_1{}^\kappa\}$ and $(s - \{\beta^\kappa\}) \cup \{\beta_2{}^\kappa\}$;
4. if $s[\nu^\kappa]$ is a path, so is $s \cup \left\{\nu_0{}^{\kappa j}\right\}$, $1 \leq j \leq \mu_M(\nu_0)$;
5. if $s[\pi^\kappa]$ is a path, so is $s \cup \{\pi_0{}^\kappa\}$.

The path $(s - \{\alpha^\kappa\}) \cup \{\alpha_1{}^\kappa, \alpha_2{}^\kappa\}$ is said to have been *obtained by reduction on* α^κ from s; similar terminology is used in the other cases.

Each path s through X^{μ_M} determines a set (or *sequent*) of positions as follows:

$$\mathcal{S}(s) = \{\, x \mid x \leq y \quad \text{for some} \quad y \in s \,\}.$$

A path s through X^{μ_M} is *atomic* iff for every k^κ in s either

(a) k is labeled by an atomic formula, or

(b) k is a ν and for all j, $1 \leq j \leq \mu_M(\nu_0)$, $\nu_0{}^{\kappa j} \in \mathcal{S}(s)$.

Remark. Our definition of path differs from Andrews' [And81] and Bibel's [Bib81] definition so as to demonstrate the relationship between the matrix methods and sequent/tableau methods. Their paths correspond roughly to our atomic paths. Each clause in our definition, when interpreted as operating on the sequent associated with the path, corresponds to an (inverted) rule of Kanger's system. Furthermore, for a given multiplicity, the sequent associated with an atomic path is *complete* in the sense that its membership cannot be increased by the application of further rules. These relationships are discussed in more detail in [Wal86]. ■

We can see the atomic paths through an indexed formula by writing the components of an α-type subformula side by side and the components of a β-type subformula one above the other to form a nested *matrix*. Multiple instances of ν_0-subformulae are treated as multiple components of an α-type parent. The matrix representation of the indexed formula of Figure 2 is shown in Figure 3. The atomic paths through the formula are then (roughly) the horizontal

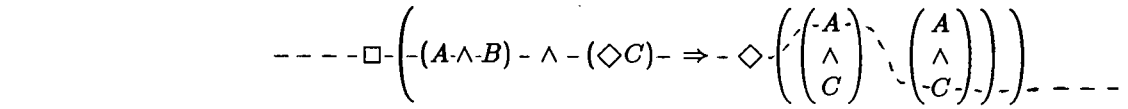


Figure 3: Matrix representation of a formula.

matrix paths that consist of atomic formulae. There are four such paths in our example. One is shown as a dotted line in the figure.

A *connection* in X^{μ_M} is an unordered pair of positions of its formula tree labeled by the *same* atomic formula but of *different* polarities. In terms of the sequent system we wish to interpret a connection as an instance of a basic sequent. A set of connections is said to *span* X^{μ_M} just when every atomic path through X^{μ_M} contains a connection from the set. For example, the connections $\left\{ a_5, a_{11}^{(1)} \right\}$ and $\left\{ a_8, a_{12}^{(1)} \right\}$ span our example formula.

3.4 Complementarity.

In order to interpret a connection as an instance of a basic sequent we must ensure that the two atomic formulae represented by the positions of a connection have the same prefix. These prefixes are determined when the modal operator dominating the atomic formula is reduced using the inverted modal rules.

Let T denote the set $\mathcal{V}_0 \cup \Pi_0$. We use $\mathrm{pre}(u)$ to denote the $<$-greatest T-element that dominates u in the formula ordering. $\mathrm{pre}(u)$ is called the *prefix* of u. Note that since the root node of a formula tree is a T-element this notion is well-defined. We shall use p, q, to denote positions when we are considering them as prefixes. In our ongoing example, the prefix of a_5 is a_2, whereas the prefix of $a_{11}^{(1)}$ is $a_{10}^{(1)}$.

The next step is to notice that we are free to choose the prefix introduced when we use the ν-rule, but must introduce arbitrary new prefixes when utilising the π-rule. We have indicated that the two positions that constitute a connection must have the same prefix. Consequently we treat ν_0-type prefixes as variables and π_0-type prefixes as constants and build a *modal substitution* $\sigma\colon \mathcal{V}_0 \to T$ under which the required prefixes are identical.

For example, the modal substitution $\sigma = \left\{ a_{10}^{(1)} \leftarrow a_2 \right\}$ renders the connection $\left\{ a_5, a_{11}^{(1)} \right\}$ complementary. Recall that a_2 is a π_0-type positon and hence a constant whereas $a_{10}^{(1)}$ is a ν_0-type position and therefore a variable. We cannot build a consistent substitution that also makes the connection $\left\{ a_8, a_{12}^{(1)} \right\}$ complementary since this would have to involve the component $a_{10}^{(1)} \leftarrow a_8$ and a_2 and a_8 are distinct constants.

The substitution of a π_0 prefix for a ν_0 prefix entails that the former is introduced in place off the latter with the ν-rule. But this means that π_0 prefixes may be introduced into a sequent before the use of the π rule that introduced them from their parent and hence the proviso on the rule would not be met. We must ensure that this never happens.

A modal substitution $\sigma\colon \mathcal{V}_0 \to T$ induces an equivalence relation $\sim_M$ and a relation $\sqsubset_M$ on $T \times T$ as follows:

1. If $\sigma(u) = v$ for some v of ν_0-type, then $u \sim_M v$.
2. If $\sigma(u) = v$ for some v of π_0-type, then $v \sqsubset_M u$.

3. If $v \sqsubset_M u$ and $u \sim_M u'$, then $v \sqsubset_M u'$.

The substitution σ is S5-*admissible* provided the reduction ordering $\lhd \stackrel{\text{def}}{=} (< \cup \sqsubset_M)^+$ is irreflexive.

The relation $v \sqsubset_M u$ between a π_0 and a ν_0 position indicates that the formula rooted at the parent of v (a π-type formula) should be reduced using the π rule to introduce the prefix v before the parent of u (a ν-type formula) is reduced using the ν rule to introduce the prefix u, the value of which is also v under the modal substitution. The equivalence relation $\sim_M$ indicates that the two ν_0-type prefixes must take the same value under the substitution.

Let σ be an S5-admissible mapping for X^{μ_M}. A connection $\{x, y\}$ in X^{μ_M} is said to be σ-*complementary* iff $\sigma(\text{pre}(x)) = \sigma(\text{pre}(y))$. A set of connections is said to be σ-complementary iff all its elements are σ-complementary.

We are now in a position to state our extension of Bibel's connection theorem to S5 modal logic.

Theorem 3.4.1 *A formula A is* S5*-valid iff there is a modal multiplicity μ_M, an* S5*-admissible mapping σ and a set of σ-complementary connections that spans $\langle A, 0\rangle^{\mu_M}$.*

We omit the proof of this theorem due to lack of space. Correctness follows directly from the correctness of Kanger's sequent system together with the argument that the irreflexivity of $\lhd$ ensures the existence of a correct sequent proof respecting the proviso on the π rule. Completeness is obtained by showing that a modal multiplicity can be constructed such that the sequent associated with an atomic path containing no complementary connection forms an S5-Hintikka set which is realizable (see [Fit83]).

4 Implementation.

An implementation of this connection calculus for S5 has been developed from an existing implementation of a connection calculus for first-order logic [Wal83]. The implementation language chosen was Quintus PROLOG since we were more concerned with the techniques used to implement the proof system than in absolute efficiency. This section describes some of the interesting features of the implementation, a more comprehensive description can be found in [Wil86].

4.1 Static data structures.

4.1.1 Formula tree representation.

Each position of the formula tree must be stored in some space-efficient way which permits a time-efficient lookup mechanism. It is important that this storage method also allow the program to manipulate portions of the formula tree without actually making copies of those portions, so that demands on storage space do not become excessive. The formula tree forms the basis for an implementation utilising structure sharing.

For both implementations (first-order and S5) each position is stored as a tuple in the PROLOG database. While there are three conceptually distinct tuple types for representing quantifiers or modal operators, connectives, and atomic formulae, the same overall structure is used throughout.

The data fields in these tuples record such things as the polarity of the position, its descendents in the formula ordering, its label (the concrete syntax at that point in the formula tree), a pointer to its parent position, and a pointer to its next sibling in the formula tree. (Recall that the label of a position will be either a connective, a quantifier or modal operator, or an atomic formula.) These tuples are constructed and placed in the database as the formula is read in, and, with one exception described in the next section, are never modified thereafter.

Each tuple is assigned a unique numeric identifier as it is constructed.

All references to tuples by the theorem prover are made through the use of pointers. A pointer consists of a numeric identifer together with an index indicating the particular instance of the tuple being referenced.

This representation allows formula tree nodes to be accessed in a variety of ways other than by reference to their identifiers. For example, all occurrences of nodes containing a particular proposition or connective, or having a particular polarity, can be retrieved using the same sort of database lookup which retrieves the tuple associated with a particular identifier.

The assignment of numeric identifiers to tuples is done depth-first and left-to-right, so as to facilitate various frequently performed operations. For instance, if n and m are two position identifiers, and n' is the next sibling of n then m is dominated by n just in case (a) $n < m$ and (b) $m < n'$ (*i.e.,* dominance can be determined using just two integer comparisons). For example, the signed modal formula $\langle \Box((A \wedge B) \wedge \Diamond C \Rightarrow \Diamond(A \wedge C)), 0\rangle$, represented by the tree shown in Figure 1, would be stored in the PROLOG database as shown in Figure 4.

Identifier	Syntax	Polarity	Type	Descendents	Next Sibling
(1,	$\Box$,	0,	π,	(2),	13)
(2,	$\Rightarrow$,	0,	α,	(3,9),	13)
(3,	$\wedge$,	1,	α,	(4,7),	9)
(4,	$\wedge$,	1,	α,	(5,6),	7)
(5,	A,	1,	–,	–,	6)
(6,	B,	1,	–,	–,	8)
(7,	$\Diamond$,	1,	π,	(8),	10)
(8,	C,	1,	–,	–,	11)
(9,	$\Diamond$,	0,	ν,	(10),	13)
(10,	$\wedge$,	0,	α,	(11,12),	13)
(11,	A,	0,	–,	–,	12)
(12,	C,	0,	–,	–,	13)

Figure 4: Example modal database.

4.1.2 Multiplicity and prefixes.

The modal multiplicity records the number of distinct instances of subformulae dominated by a ν type modal operator allowed within the (partial) proof at any given point in time. Positions are always considered indexed as described in section 3.1. The modal multiplicity is represented as a table with one entry for each ν_0 position in the formula tree. Indices are therefore stored as lists of positive integers. For S5, prefixes are simply positions which represent modal operators.

4.1.3 Proof Tree Representation

The proof state within a connection calculus is represented by a record of the connections made so far, the substitution constructed and the paths through the formula still to checked for the current multiplicity. OR choices arise when there is more than one possible connection that makes a given path complementary (see Section 4.2.1). The OR choices are maintained as a tree.

In this implementation, each node of the tree is stored in the database as it is constructed and identified by a string of bits. The identifier for a node N is constructed according to the following rules:

1. The identifier of the root node is 1.

2. If N is the right descendent of a node M and the identifier of M is B, the identifier of N is $1B$.

3. If N is the left descendent of a node M (or only descendent, if M is a unary node), and the identifier of M is B, the identifier of N is $0B$.

When these rules have been followed, the node whose identifer is I_1 is an ancestor of the node whose identifier is I_2 iff I_1 is a right-adjusted substring of I_2. This test can be done directly on the identifiers themselves, without requiring any database lookups or tree traversals. However, this test is more complicated, and hence slower, than the corresponding integer comparison test used to determine ancestry of formula tree nodes.

Storage of node identifiers is also more complicated for the dynamic proof trees than for the static formula trees. A node at depth D requires a bit-string D bits long to identify it. For trees of reasonable depth, D exceeds the length of the standard PROLOG integer, and so the bit string must be split across several such integers. In practise, identifiers and pointers are recorded using a three-place data structure, "`code(X,Y,Z)`", in which "Z" is a list of 16-bit integers, and "X" and "Y" identify the word and bit position containing the bit most recently added to the identifer.

Considerable savings in space could have been obtained if the search strategy was confined to depth-first, left-to-right through the space of possible connections. (Such a procedure resembles the search strategy of standard PROLOG interpreters.) However the original program was designed to investigate aspects of heuristic search, so a more flexible representation of partial proofs was required.

4.1.4 Connection graphs.

The last static data structure of interest is the *connection graph*. Since only a single copy of the formula tree is kept, the positions in which a particular atomic formula may be found with a particular polarity can be tabulated as the formulae are being input. This table is called a "connection graph", and by referring to it while constructing proofs the theorem prover never needs to search the formula tree to find complementary propositions. The connection graph is propositional but could be extended to include unification information.

Note that the use of a connection graph is distinct from the use of connection graph ***resolution*** [Kow75] in which alterations to this data structure are a part of the inference mechanism. The connection graph used here is not manipulated or changed in any way once the proof process is initiated.

Remark. However the two systems are in some interesting sense complementary. Connection graph resolution could be used to make permanent alterations in the formula tree so as to "compile" certain inferences and cut down on search within a run of the connection method. To our knowledge, this potential has not been exploited. ■

4.2 Dynamic features.

4.2.1 Path checking and goal representation.

The connection theorem (in the case of S5, Theorem 3.4.1) underlying a given connection calculus expresses the validity of a formula in terms of a condition on the set of atomic paths through the formula. The main component of that condition is that a set of connections can be found such that every atomic path contains at least one connection from that set. Checking a formula for validity therefore reduces in part to a process of *path checking*. Indeed, many resolution based proof procedures can be interpreted in this way and a comparison made between them [Bib82b].

This path checking process can best be expressed using the *matrix* representation of the formula introduced above in which the components of a β-type formula are displayed one above the other in the plane whilst the components of an α-type formula are displayed side by side.

The algorithm we use to check the atomic paths through a formula is based on that presented in [Bib82a]. A partial proof is represented by a set of *goals*. Each goal represents a set of atomic paths which have not yet been fully examined, but which share the same initial segment. The initial segment itself is divided into two parts, the *active path* and the *expansion*. The active path is a set of atomic formulae amongst which no connections exist. When choosing between possible connections, those involving the atomic formulae referenced in the active path are preferred over all others. This is equivalent to the unit preference heuristic [CL73]. The expansion is a set of references to partially examined subformulae. These references are generated by the reduction of α-type nodes, and correspond to dynamically created hypotheses which may contain extra problem-specific information in the form of variable instantiations not present in the original axioms. When choosing between possible connections, those involving the subformulae referenced in the expansion are preferred over those involving uninstantiated axioms. This is similar to the "set-of-support" heuristic used in resolution based systems.

4.2.2 Unification.

The reduction ordering is represented by a graph whose nodes are prefixes and whose arcs indicate precedence and/or equivalence of nodes. When a connection is formed, and two prefixes are unified, one new arc is added to this graph to show the effect on reduction order of the unification, and zero or more arcs added to show any new precedence information introduced by formula ordering. (Recall that the reduction ordering $\lhd$ is the transitive closure of the union of the substitution relation and the formula ordering.) If at any point this graph becomes cyclic as a result of the introduction of arcs and/or nodes to represent a particular unification, the connection inducing that unification is inadmissible.

The unification of prefixes in S5 is equivalent to unification in a subset of first-order logic containing only constants and variables, and not functions or terms. Consequently, the representation used for prefixes, and the way in which they are unified, can be simpler than the corresponding mechanisms within the implementation for first-order logic. For example, a substitution table must be maintained when performing unification in first-order logic to allow the two unifications $x \leftrightarrow f(a,b)$ and $x \leftrightarrow f(b,a)$ to be distinguished. No such table is needed when unifying prefixes in S5 on the other hand, because all such unification is one-to-one. Similarly, unification of S5 prefixes is non-recursive, since there is never any need to "unpack" a term so as to unify its subterms.

4.2.3 Reporting proof search.

In [Wal86] it was shown how a connection-based search for first-order logic could be interpreted as constructing a generalised form of sequent (or tableau) proof. The derivations are "generalised" because the choice of prefix at the reduction of a ν type position is delayed until a later date via the process of unification. The derivations are similar in spirit to those constructed using Jackson and Reichgelt's sequent systems [JR87], or modal versions of Reeves' tableau system [Ree87].

In the implementation, this correspondence was exploited by reporting the search for connections as a sequence of applications of Kanger's sequent rules.

Remark. It is important to realize that the search is conducted at the level of connections. The use of sequent rules for proof output is simply a matter of "pretty-printing" to provide informative output. This should be contrasted with sequent based systems that search a much larger space, full of redundancies, in order to obtain such "natural" reporting facilities [BT75].

■

While this is a powerful result, many problems remain. We shall mention two.

Firstly, the naive translation of connections into sequent derivations (partial proof trees) does not produce an intuitively satisfying sequent proof (even though this proof is correct). The problem lies in the fact that although each individual connection can be reported as an intelligible derivation, the juxtaposition of derivations corresponding to successive connections does not necessarily result in any recognisable pattern.

Attempts were made to change the order in which reduction steps were displayed so as to make the resulting output more comprehensible, including printing some sequences bottom-up and others top-down, but no simple solution was found. Some *ad hoc* mechanisms incorporated into the theorem prover were useful. For example, when a position representing an implication is reduced, the theorem prover will print either

$$\texttt{Rule}: -> \texttt{C} \qquad \text{or} \qquad \texttt{Rule}: \texttt{A} -->$$

depending on whether the consequent or antecedent branch of the formula tree is being pursued.

Work by Andrews and his colleagues [And80,Mil84] may prove useful in this context even though these authors are concerned with the presentation of a *complete* proof expressed in terms of connections as a natural deduction or sequent calculus proof. Here we are concerned with the reporting the proof search directly.

The second problem concerns S5 and modal logics in general. For these logics even well-formed sequent proofs are not intelligible since propositions may be true in one context and false in another; sequent proofs are not appropriate for the display of these contexts. A display facility based on the box diagrams of [HC68] is one possible approach.

5 Related work.

A number of authors have developed computationally oriented proof systems for S5. Some are based on clausal resolution (*eg.,* [Far83]) since S5 does admit a modal clausal form. Bibel's comparison of the classical connection calculus with classical clausal techniques in [Bib82b] suffice to demonstrate the benefits of our approach.

Abadi and Manna [AM86] develop a system based on non-clausal resolution. Their system suffers from various combinatorial problems due to the fact that the modal operators are manipulated by special deduction rules. The application of resolution is severely restricted. Our use of prefixes and unification removes the need for the special rules and liberates the use of the basic resolution operation (making a connection).

Konolige system [Kon86] involves the construction of multiple tableaux. Resolution is restricted to operate within each tableau. Again the combinatorics of his system are not ideal; indeed the construction of an auxiliary tableau is only justified in retrospect introducing many sources of redundancy. Our use of prefixes eliminates the need to consider multiple tableaux.

6 Conclusions.

We have presented the theory and some details of an implementation of a computationally efficient proof system for S5 modal logic. The system is a variant on Bibel's connection calculus for first-order logic. The implementation was developed in a straightforward manner from a previous implementation of the connection calculus for first-order logic.

The basic techniques, summarised at the beginning of Section 3.1 were:

> The search should be confined in the first instance to selecting instances of the basic sequent schema of the appropriate sequent calculus. A process of unification together with a reduction ordering should be used to ensure that a correct sequent proof could be constructed at every stage without overcommitting the system to choices of particular values for the "variable" constructs. (In the case of first order systems these variable constructs are universally quantified variables. In the modal case the constructs are prefixes denoting possible worlds.)

It was suggested in [Wal86] that the techniques applied here to what is in effect the simplest modal logic could be used to produce computationally efficient proof systems for more complicated logics. We have succeeded in constructing a generalised connection calculus for the modal logics K, K4, D, D4, T, S4 and S5, as well as both constant- and varying-domain versions of their first-order variants. The details can be found in [Wal87]. In the latter case we can deal with versions of the logics in which the so-called "monotonicity" condition holds between worlds and versions in which no such condition holds. The generalised connection calculus is specialised to a particular version of a particular logic by altering the admissibility conditions on the modal substitution. All other details (including the methods for proof search) remain the same.

Acknowledgements

This research was supported in part by SERC/Alvey grants GR/D/44874 and GR/D/44270, and a British Council Commonwealth scholarship to the second author. We are indebted to Richard O'Keefe who described the utility of the numbering method for formula trees adopted, and Jane Hesketh who helped develop the bit string representation of integers in PROLOG.

References

[AM86] M. Abadi and Z. Manna. Modal theorem proving. In J.H. Siekmann, editor, *8th International Conference on Automated Deduction*, pages 172–189, July 1986. Lecture Notes in Computer Science, Volume 230, Springer Verlag.

[And80] P.B. Andrews. Transforming matings into natural deduction proofs. In W. Bibel and R. Kowalski, editors, *5th International Conference on Automated Deduction*, pages 281–292, 1980. Lecture Notes in Computer Science, Volume 87, Springer Verlag.

[And81] P.B. Andrews. Theorem-proving via general matings. *Journal of the Association for Computing Machinery*, 28(2):193–214, April 1981.

[Bib81] W. Bibel. On matrices with connections. *Journal of the Association for Computing Machinery*, 28(4):633–645, October 1981.

[Bib82a] W. Bibel. *Automated Theorem Proving.* Friedr. Vieweg & Sohn, Braunschweig, 1982.

[Bib82b] W. Bibel. A comparative study of several proof procedures. *Artificial Intelligence*, 18:269–293, 1982.

[BT75] W.W. Bledsoe and M. Tyson. *The UT Interactive Prover.* Research Report ATP-17, Departments of Mathematics and Computer Sciences, University of Texas at Austin, May 1975.

[CL73] C-L. Chang and R.C-T. Lee. *Symbolic logic and mechanical theorem proving.* Academic Press, 1973.

[Far83] L. Fariñas-del-Cerro. Temporal reasoning and termination of programs. In S. Amarel, editor, *8th International Joint Conference on Artificial Intelligence*, pages 926–929, 1983.

[Fit72] M.C. Fitting. Tableau methods of proof for modal logics. *Notre Dame Journal of Formal Logic*, XIII:237–247, 1972.

[Fit83] M.C. Fitting. *Proof methods for modal and intuitionistic logics.* Volume 169 of *Synthese library*, D. Reidel, Dordrecht, Holland, 1983.

[HC68] G.E. Hughes and M.J. Cresswell. *An Introduction to Modal Logic.* Methuen, Londo 1968.

[HM84] J.Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. In *3rd ACM Conference on the Principles of Distributed Computing*, pages 50–61, 1984.

[HM85] J.Y. Halpern and Y. Moses. A guide to the modal logics of knowledge and belief:preliminary draft. In *9th International Joint Conference on Artificial Intelligence*, pages 479–490, 1985.

[JR87] P. Jackson and H. Reichgelt. A general proof method for first-order modal logic. Submitted to IJCAI-87, 1987.

[Kan57] S. Kanger. *Provability in logic.* Volume 1 of *Stockholm Studies in Philosophy*, Almqvist and Wiksell, Stockholm, 1957.

[Kon84] K. Konolige. *A Deduction Model of Belief and its Logics.* PhD thesis, Stanford University, 1984.

[Kon86] K. Konolige. Resolution and quantified epistemic logics. In J.H. Siekmann, editor, *8th International Conference on Automated Deduction*, pages 199–208, July 1986. Lecture Notes in Computer Science, Volume 230, Springer Verlag.

[Kow75] R. Kowalski. A proof procedure using connection graphs. *Journal of the Associatic for Computing Machinery*, 22(4):572–595, 1975.

[Mil84] D.A. Miller. Expansion tree proofs and their conversion to natural deduction proo In R.E. Shostak, editor, *7th International Conference on Automated Deductic* pages 375–393, May 1984. Lecture Notes in Computer Science, Volume 170, Spring Verlag.

[Moo80] R.C. Moore. *Reasoning about knowledge and action.* Technical Note 191, SRI International, Menlo Park, Ca., 1980.

[Pne77] A. Pneuli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*, pages 46–57, 1977.

[Ree87] S. Reeves. Semantic tableaux as a framework for automated theorem proving. In J. Hallam and C. Mellish, editors, *(This proceedings)*, Wiley & Sons, 1987.

[Smu68] R.M. Smullyan. *First-Order Logic.* Volume 43 of *Ergebnisse der Mathematik*, Springer-Verlag, Berlin, 1968.

[Sti85] C. Stirling. *Modal logics for communicating systems.* Technical Report CSR-193-85, Dept. of Computer Science, Edinburgh University, 1985.

[Wal83] L.A. Wallen. *Towards the Provision of a Natural Mechanism for Expressing Domain-Specific Global Strategies in General Purpose Theorem-Provers.* Research Paper 202, Dept. of Artificial Intelligence, Edinburgh, September 1983.

[Wal86] L.A. Wallen. Generating connection calculi from tableau- and sequent-based proof systems. In A.G. Cohn and J.R. Thomas, editors, *Artificial Intelligence and its Applications*, pages 35–50, Wiley & Sons Ltd., 1986.

[Wal87] L.A. Wallen. Matrix proof methods for modal logics. In J. McDermott, editor, *10th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann Inc., 1987. To appear.

[Wil86] G.V. Wilson. *Implementation of a connection method theorem-prover for S5 modal logic.* Master's thesis, Department of Artificial Intelligence, University of Edinburgh, 1986.

Matrix proof methods for modal logics

Lincoln A. Wallen
Dept. of Artificial Intelligence
University of Edinburgh
Scotland

Abstract

We present matrix proof systems for both constant- and varying-domain versions of the first-order modal logics K, K4, D, D4, T, S4 and S5 based on modal versions of Herbrand's Theorem specifically formulated to support efficient automated proof search. The systems treat the full modal language (no normal-forming) and admit straightforward implementations using structure-sharing techniques. A key feature of our approach is the use of a specialised unification algorithm to reflect the conditions on the accessibility relation for a given logic. The matrix system for one logic differs from the matrix system for another only in the nature of this unification algorithm. In addition, proof search may be interpreted as constructing generalised proof trees in an appropriate tableau- or sequent-based proof system. This facilitates the use of the matrix systems within interactive environments.

1 Introduction.

Modal logics are widely used in various branches of artificial intelligence and computer science as logics of knowledge and belief (*eg.,* [Moo80,HM85,Kon84]), logics of programs (*eg.,* [Pne77]), and for specifying distributed and concurrent systems (*eg.,* [HM84,Sti85b]). As a consequence, the need arises for proof systems for these logics which facilitate efficient automated proof search.

Traditional proof systems for modal logics, such as tableau- or sequent-based systems are readily available (*eg.,* [Kan57,Nis83,Fit83]). While these systems are to some extent human-oriented, the proof rules form an inadequate basis for automated proof search since they generate search spaces that contain considerable redundancies. The redundancies arise mainly from the characteristic emphasis on connectives and the proof rules for modal operators and quantifiers.

The matrix methods for first-order classical logic, pioneered by Prawitz [Pra60], and further developed by Andrews [And81] and Bibel [Bib81], have been demonstrated to be less redundant than the most efficient of the resolution based methods for that logic [Bib82b]. The methods combine an emphasis on *connections* (drawn from the resolution methods) with an intensional notion of a *path.*

In this paper we present matrix proof systems for the modal logics K, K4, D, D4, T, S4 and S5, based on modal versions of Bibel's "computationally improved" Herbrand Theorem for first-order classical logic [Bib82c]. We consider both constant- and varying-domain versions of the first-order modal logics.

The major features of our approach may be summarised as follows. Validity within a logic is characterised by the existence of a set of connections (pairs of atomic formula occurrences: one positive, one negative) within the formula, with the property that every so-called atomic path through the formula contains (as a subpath) a connection from the set (§ 2.4). Such a set of connections

Proceedings of the 10th International Joint Conference in Artificial Intelligence, pages 917–923, Morgan Kaufmann Inc., 1987.

is said to span the formula. For classical propositional logic this condition suffices [And81,Bib81]. For first-order logic a substitution (of parameters or terms for variables) must be found under which the (then propositional) connections in the spanning set are simultaneously complementary. Conditions are placed on the substitution that ensure amongst other things that a proof within a particular tableau- or sequent-based proof system is constructable from the connections and the substitution [Bib82c,Wal86]. This basically amounts to ensuring that the restrictions found on the traditional quantifier rules can be met.

For the propositional modal logics we keep the basic matrix framework but define a notion of complementarity for atomic formulae that ensures the existence of a proof in one of Fitting's *prefixed* tableau systems [Fit72,Fit83]. This amounts to ensuring that, semantically: the two atomic formulae of a connection can be interpreted as inhabiting the same "possible world," and proof-theoretically: that they can be given the same prefix (§ 2.5.1). The key observation is that this can be established by noting the position of the atoms relative to the modal operators in the original formula and utilising a specialised unification algorithm operating over representations of these positions. Clearly, this notion of complementarity is logic-dependent, a dependence which is reflected in the choice of unification algorithm. Lifting these results to first-order constant-domain modal logics is simply a matter of combining this modal notion of complementarity with the first-order notion (§ 2.5.2).

For the varying-domain versions we index individual variables with the prefix of their quantifier. The substitution of one variable for another is permitted provided their prefixes can be unified (§ 2.5.2).

Checking a formula for validity within a modal logic is therefore reduced to a process of path checking and complementarity tests performed by a specialised unification algorithm (§ 3). During this process extra copies may need to be considered of universally quantified formulae and/or formulae dominated by a modal operator of "necessary" ($\Box$) force. The duplication in both cases is managed by an extension of Bibel's indexing technique or *multiplicity* [Bib82a] which supports the implementation of the matrix systems using structure-sharing techniques [BM72]. The notions of multiplicity, substitution and spanning sets of connections form the basis of the relationship with Herbrand's Theorem.

A number of authors have attempted to adapt computationally oriented proof systems for first-order logic to the modal logics considered here (*eg.*, [Far83], [AM86a], [Kon86]). We compare our approach favourably to theirs in Section 4.

2 The modal matrix systems.

2.1 Preliminaries.

We assume familiarity with the usual definition of the modal language and formulae. We let A, B range over formulae and P, Q range over atomic formulae.

A pair $\langle G, R\rangle$, comprising a non-empty set G and a binary relation R on G is called a *frame*. Let D be some non-empty set. A *first-order frame over* D is a triple $\langle G, R, P\rangle$ where $\langle G, R\rangle$ is a frame and P is a mapping from G to non-empty subsets of D. $P(w)$ can be interpreted as the set of individuals that "exist" in the world w.

We can obtain different versions of the first-order logics by restricting the way in which P varies from world to world. For example, we could require the *constant-domain* condition: for $w, w' \in G$, $P(w) = P(w')$. Axiomatically, constant-domain modal logics are obtained by including the so-called Barcan formula $\forall x \Box Ax \Rightarrow \Box \forall x Ax$ as an additional axiom. Our purpose here is not to choose between these possibilities but to develop matrix proof systems for each of the variants.

If we restrict R to satisfy the conditions outlined in Table 1, we say that $\langle G, R, P\rangle$ is an $\mathcal{L}$-frame over D, where $\mathcal{L}$ is the logic associated with the conditions. The "idealization" condition is that

$\mathcal{L}$	Condition on R
K	no conditions
K4	transitive
D	idealization
D4	idealization, transitive
T	reflexive
S4	reflexive, transitive
S5	equivalence

Table 1: Conditions on accessibility relations.

for every element $w \in G$ there is some element $w' \in G$ such that $w\ R\ w'$. Once again our purpose is not to choose between these logics but to develop matrix proof systems for each.

An $\mathcal{L}$-model over D is a pair $\langle\langle G, R, P\rangle, \Vdash\rangle$ where $\langle G, R, P\rangle$ is an $\mathcal{L}$-frame over D and $\Vdash$ is a relation between elements of G and sentences such that: for all $w \in G$

1. $w \Vdash A \wedge B$ iff $w \Vdash A$ and $w \Vdash B$.
2. $w \Vdash A \vee B$ iff either $w \Vdash A$ or $w \Vdash B$.
3. $w \Vdash A \Rightarrow B$ iff either $w \nVdash A$ or $w \Vdash B$.
4. $w \Vdash \neg A$ iff $w \nVdash A$.
5. $w \Vdash \Box A$ iff for all $v \in G$ with $w\ R\ v$, $v \Vdash A$.
6. $w \Vdash \Diamond A$ iff for some $v \in G$ with $w\ R\ v$, $v \Vdash A$.
7. $w \Vdash \forall x A$ iff for all $d \in P(w)$, $w \Vdash A[d/x]$.
8. $w \Vdash \exists x A$ iff for some $d \in P(w)$, $w \Vdash A[d/x]$.

Satisfaction in a model and validity are defined as usual.

A *signed* formula is a pair $\langle A, n\rangle$ where A is a formula and $n \in \{0, 1\}$. We let $\boldsymbol{X}, \boldsymbol{Y}$ range over signed formulae. Informally, the signs "1" and "0" should be interpreted as the qualifiers "is true" and "is false" respectively. For ease of exposition we use a uniform notation due to Smullyan and Fitting that classifies signed formulae according to their sign and major connective/operator as shown in Table 2.

2.2 Formula occurrences.

A *formula tree* for a signed formula is a variant of its formation tree containing additional information as to the polarity of its subformula occurrences (*i.e.*, whether an occurrence of a subformula is negative or positive within the formula). It is best explained by example. A formula tree for the signed formula $\langle \Diamond\Diamond\forall x\,(\Diamond Px \wedge \Box Qx) \Rightarrow \Diamond(\forall y Py \wedge \forall z Qz), 0\rangle$ is shown in Figure 1. Following Bibel [Bib82c], we name the nodes or *positions* of the tree (a_0-a_{14}) so as to distinguish different occurrences of the same subformula. With each position we associate the polarity of the subformula rooted at that position and a *label* consisting of the major connective/operator of that subformula, or the subformula itself when that is atomic. We use $<$ to denote the (partial) ordering in the formula tree. Positions form the basis for implementations of the matrix systems using structure-sharing techniques [BM72]. A position should be interpreted as a pointer to a single copy of the main formula stored in computer memory. Unlike resolution-based methods, these matrix methods do not require the explicit generation of intermediate formulae.

α	α_1	α_2
$\langle A \wedge B, 1\rangle$	$\langle A, 1\rangle$	$\langle B, 1\rangle$
$\langle A \vee B, 0\rangle$	$\langle A, 0\rangle$	$\langle B, 0\rangle$
$\langle A \Rightarrow B, 0\rangle$	$\langle A, 1\rangle$	$\langle B, 0\rangle$
$\langle \neg A, 1\rangle$	$\langle A, 0\rangle$	$\langle A, 0\rangle$
$\langle \neg A, 0\rangle$	$\langle A, 1\rangle$	$\langle A, 1\rangle$

ν	ν_0
$\langle \Box A, 1\rangle$	$\langle A, 1\rangle$
$\langle \Diamond A, 0\rangle$	$\langle A, 0\rangle$

γ	γ_0
$\langle \forall x A, 1\rangle$	$\langle A, 1\rangle$
$\langle \exists x A, 0\rangle$	$\langle A, 0\rangle$

β	β_1	β_2
$\langle A \wedge B, 0\rangle$	$\langle A, 0\rangle$	$\langle B, 0\rangle$
$\langle A \vee B, 1\rangle$	$\langle A, 1\rangle$	$\langle B, 1\rangle$
$\langle A \Rightarrow B, 1\rangle$	$\langle A, 0\rangle$	$\langle B, 1\rangle$

π	π_0
$\langle \Box A, 0\rangle$	$\langle A, 0\rangle$
$\langle \Diamond A, 1\rangle$	$\langle A, 1\rangle$

δ	δ_0
$\langle \forall x A, 0\rangle$	$\langle A, 0\rangle$
$\langle \exists x A, 1\rangle$	$\langle A, 1\rangle$

Table 2: Classification of signed formulae.

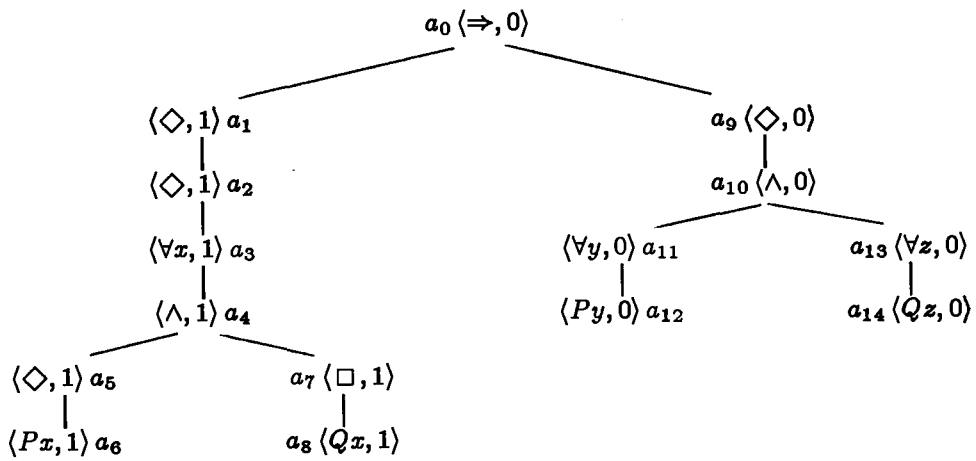


Figure 1: Formula tree for $\langle \Diamond\Diamond\forall x\,(\Diamond Px \wedge \Box Qx) \Rightarrow \Diamond(\forall y Py \wedge \forall z Qz), 0\rangle$.

In terms of analytic tableau systems [Smu68,Fit83], the polarity of a position determines the sign with which the subformula rooted at that position will occur on any analytic tableau which has the main (signed) formula as root [Wal86]. Therefore, the classification of signed formulae can be extended to the positions of a formula tree by defining the type of a position to be the type of the signed formula rooted at that position in the tree. In addition, we consider the root node of the formula tree to be of π_0 type.

Notice that each position has two types: its principal type (*eg.*, $\alpha, \beta, \nu, \ldots$) is determined by its label and polarity, while its secondary type (*eg.*, $\alpha_1, \alpha_2, \beta_1, \ldots$) arises from the type of its parent.

For a given formula tree we use k, l, possibly subscripted, to denote positions and $\mathcal{V}_0$, Π_0, Γ_0 and Δ_0 to denote the sets of positions of type ν_0, π_0, γ_0 and δ_0 respectively.

2.3 Multiplicities.

The semantic clauses for interpreting formulae whose major symbol is a modal operator of "necessary" force (*i.e.*, ν-type) or a quantifier of "universal" force (*i.e.*, γ-type) indicate that we must consider multiple instances of the principle subformula of such a formula occurrence (*i.e.*, formulae rooted at ν_0- and γ_0-type positions respectively within a formula tree). In the modal case we intend different instances to inhabit different worlds; in the case of a quantified formula, we form different instances by the substitition of different parameters for the (universally) bound variable.

The following definitions are introduced for a given formula tree for a given signed formula X.

A function μ_M from $\mathcal{V}_0$ to the positive integers is called a *modal multiplicity* for X; it serves to encode the number of instances of subformulae of X in the scope of a modal operator of necessary force considered within a putative proof.

A function μ_Q from Γ_0 to the positive integers is called a *first-order multiplicity* for X; it serves to encode the number of instances of subformulae of X in the scope of a quantifier of universal force considered within a putative proof.

A *multiplicity* μ for X is the combination of a modal and first-order multiplicity thus: for a position k of the formula tree

$$\mu(k) = \begin{cases} \mu_M(k), & k \in \mathcal{V}_0; \\ \mu_Q(k), & k \in \Gamma_0; \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

If μ is a multiplicity for X we define the (indexed) formula tree for the *indexed formula* X^μ as a tree of indexed positions of the form k^κ, where k is a position of the basic formula tree for X and κ is a sequence of positive integers defined as follows: if $k_1 < k_2 < \cdots < k_n \leq k$, $1 \leq n$, are those ν_0- and γ_0-type positions that dominate k in the basic formula tree for X, then

$$\kappa \in \{ (j_1\, j_2 \cdots j_n) \mid 1 \leq j_i \leq \mu(k_i),\ 1 < i \leq n \}.$$

The ordering in the indexed tree $<^\mu$ is defined in terms of the ordering on the underlying tree: for indexed positions k^κ and l^τ

$$k^\kappa <^\mu l^\tau \quad \text{iff} \quad k < l \quad \text{and} \quad \tau = \kappa\theta,$$

where θ is some sequence of positive integers. The polarity and label of an indexed position k^κ is taken to be the same as the polarity and label of its underlying position k except that, in the case of atomic formulae, individual variables are indexed with the index of the child of their quantifier position (*i.e.*, a γ_0 or δ_0 position) so as to distinguish the different instances. Consequently, indexed positions inherit the type of their underlying position also.

Figure 2 shows the indexed formula tree for the example formula of Figure 1 with a multiplicity of $\mu_Q(a_4) = 2$ and constant (*i.e.*, 1) otherwise. As a convention we omit indices consisting of the empty sequence.

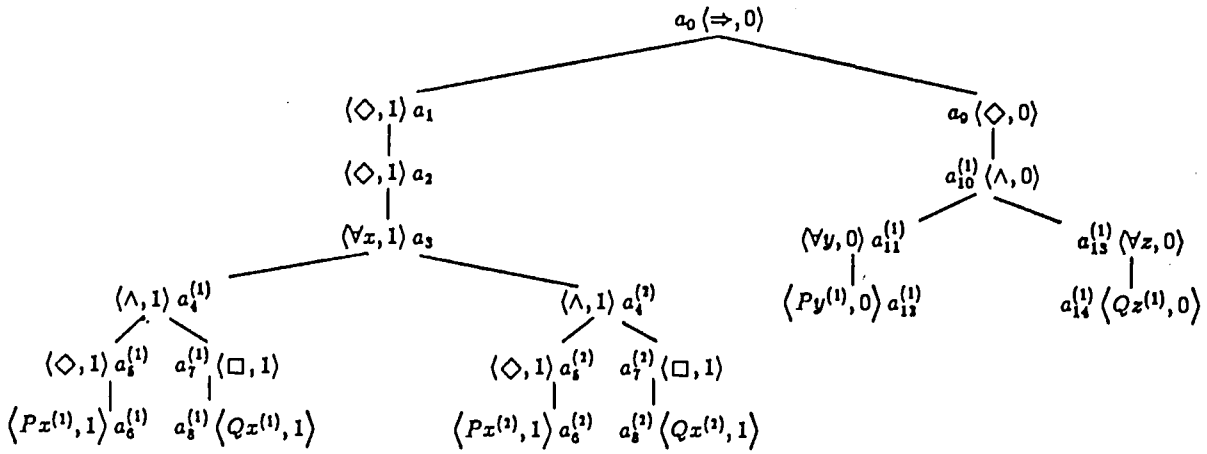


Figure 2: Indexed formula tree

We let u, v, possibly subscripted, range over indexed positions when we are not interested in the index, and drop the superscript on $<$. We abuse our notation and let $\mathcal{V}_0$, Π_0 *etc*, denote the sets of indexed positions of an indexed formula tree of the appropriate types. Henceforth we shall refer to indexed positions simply as positions.

Remark. Bibel's notion of a multiplicity [Bib82a] corresponds to our notion of a first-order multiplicity. We have altered his definition slightly to support the symmetry between the treatment of modal operators and quantifiers obtained above. Notice that, for an indexed formula, the set Γ_0 and the set of distinct universally quantified variables, and the set Δ_0 and the set of distinct existentially quantified variables in the formula are in 1-1 correspondence. We shall make use of this observation in the sequel. ■

2.4 Paths and connections.

Let X^μ be an indexed formula. A *path* through X^μ is a subset of the positions of its formula tree defined below. We shall use s, t, possibly subscripted, to denote paths, and adopt the notation $s[\alpha^\kappa]$ to denote a path s with an occurrence of a distinguished α-type position with index κ. Similarly for the other types. The set of paths through X^μ, is the smallest set such that:

1. $\left\{ k_0^{()} \right\}$ is a path, where $k_0{}^{()}$ is the root position of the formula tree for X^μ;
2. if $s[\alpha^\kappa]$ is a path, so is $(s - \{\alpha^\kappa\}) \cup \{\alpha_1{}^\kappa, \alpha_2{}^\kappa\}$;
3. if $s[\beta^\kappa]$ is a path, so are $(s - \{\beta^\kappa\}) \cup \{\beta_1{}^\kappa\}$ and $(s - \{\beta^\kappa\}) \cup \{\beta_2{}^\kappa\}$;
4. if $s[\nu^\kappa]$ is a path, so is $s \cup \left\{\nu_0{}^{\kappa j}\right\}$, $1 \leq j \leq \mu_M(\nu_0)$;
5. if $s[\pi^\kappa]$ is a path, so is $s \cup \{\pi_0{}^\kappa\}$.
6. if $s[\gamma^\kappa]$ is a path, so is $s \cup \left\{\gamma_0{}^{\kappa j}\right\}$, $1 \leq j \leq \mu_Q(\gamma_0)$;
7. if $s[\delta^\kappa]$ is a path, so is $s \cup \{\delta_0{}^\kappa\}$.

The path $(s - \{\alpha^\kappa\}) \cup \{\alpha_1{}^\kappa, \alpha_2{}^\kappa\}$ is said to have been *obtained by reduction on* α^κ from $s[\alpha^\kappa]$. Similarly in the other cases.

Each path s through X determines a set (*branch* or *sequent*) of positions as follows

$$S(s) = \{ x \mid x \leq y \quad \text{for some} \quad y \in s \}.$$

A path, s, through X^μ is an *atomic path* iff for $k^\kappa \in s$, either

(a) k is labelled by an atomic formula; or

(b) $k \in \mathcal{V}$, and for all j, $1 \leq j \leq \mu_M(\nu_0)$, $\nu_0{}^{\kappa\, j} \in S(s)$; or

(c) $k \in \Gamma$, and for all j, $1 \leq j \leq \mu_Q(\gamma_0)$, $\gamma_0{}^{\kappa\, j} \in S(s)$.

Remark. Our definition of path differs from Andrews' [And81] and Bibel's [Bib81] definition so as to demonstrate the relationship between the matrix methods and tableau/sequent methods. Each clause in the definition, when interpreted as operating on the branch associated with the path, corresponds to an analytic tableau rule [Smu68,Fit83]. A path is a representation of the *unused* formulae on a branch. Furthermore, for a given multiplicity, the branch associated with an atomic path is *complete*. These relationships are discussed in more detail in [Wal86]. ■

Consider our example (signed) formula:

$$\langle \Diamond\Diamond \forall x \, (\Diamond Px \wedge \Box Qx) \Rightarrow \Diamond(\forall y Py \wedge \forall z Qz), 0 \rangle$$

indexed as in Figure 2. If we distinguish its α-type subformulae from its β-type subformulae by placing the components of the former side-by-side and the components of the latter one above the other, we obtain a nested *matrix* thus:

$$\Diamond\Diamond\forall x \left(\left(\Diamond \left(Px^{(1)}\right) \wedge \Box \left(Qx^{(1)}\right) \right) \left(\Diamond \left(Px^{(2)}\right) \wedge \Box \left(Qx^{(2)}\right) \right) \right) \Rightarrow {}^{\Diamond} \begin{pmatrix} \forall y \left(Py^{(1)}\right) \\ \wedge \\ \forall z \left(Qz^{(1)}\right) \end{pmatrix}$$

Notice that the two instances of the subformula $Px \wedge Qx$ are considered to be the components of an implicit α-type formula. This follows from the γ clause (6) of the definition of paths above. If we omit the connectives and operators we are left with the skeleton matrix:

$$\left(Px^{(1)} \quad Qx^{(1)}\right) \quad \left(Px^{(2)} \quad Qx^{(2)}\right) \qquad \begin{pmatrix} Py^{(1)} \\ \\ Qz^{(1)} \end{pmatrix}$$

which corresponds in part to the so-called "deep formula" in the expansion tree approach of Miller [Mil84].

The atomic elements of an atomic path are simply the horizontal matrix paths through such a matrix. In this case there are two atomic paths through the formula, one with atomic elements $\{ Px^{(1)}, Qx^{(1)}, Px^{(2)}, Qx^{(2)}, Py^{(1)} \}$ and one with elements $\{ Px^{(1)}, Qx^{(1)}, Px^{(2)}, Qx^{(2)}, Qz^{(1)} \}$. More precisely, we should express these sets as positions thus:

$$\left\{ a_6^{(1)}, a_8^{(1)}, a_6^{(2)}, a_8^{(2)}, a_{12}^{(1)} \right\} \quad \text{and} \quad \left\{ a_6^{(1)}, a_8^{(1)}, a_6^{(2)}, a_8^{(2)}, a_{14}^{(1)} \right\}.$$

A *connection* in an (indexed) formula is a subpath of a path through the formula consisting of two positions labelled by an atomic formula with the same predicate symbol but of different polarities. A set of connections is said to *span* the formula just when every atomic path through it contains a connection from the set.

For example, the two connections $\left\{ a_6^{(1)}, a_{12}^{(1)} \right\}$ and $\left\{ a_8^{(1)}, a_{14}^{(1)} \right\}$ span the indexed formula displayed above. So does the connection pair $\left\{ a_6^{(1)}, a_{12}^{(1)} \right\}$ and $\left\{ a_8^{(2)}, a_{14}^{(1)} \right\}$.

2.5 Complementarity.

As remarked above, for a given multiplicity, the atomic paths through an indexed formula serve to represent the branches of a complete analytic tableau with the main formula at its root. In the same spirit, we wish to interpret connections as the two formula occurrences that atomically close the branches on which they occur.

For propositional logic, connections are complementary *by definition*. Since there is no need for multiplicities (no modal operators or quantifiers) this observation leads to a simple characterisation of validity.

Theorem 2.5.1 (Andrews [And81], Bibel [Bib81]) *A propositional formula A is valid iff there exists a set of connections that spans $\langle A, 0\rangle$.*

This theorem is the matrix counterpart to the following theorem for analytic tableaux:

Theorem 2.5.2 (Smullyan [Smu68]) *A propositional formula A is valid iff there exists an atomically closed analytic tableau for $\langle A, 0\rangle$.*

The matrix theorem is more appropriate as a basis for automated proof search because there is no need to actually construct a tableau. The spanning condition simply ensures that a tableau of the appropriate form can be constructed; we search for a spanning set of connections directly rather than via the connective oriented tableau rules [Wal86].

In the presence of modal operators and quantifiers we must be more careful. We deal with modal operators first.

2.5.1 Propositional modal systems.

Informally we must ensure that the two atomic formulae represented by the positions of a connection can be considered to inhabit the *same* possible world. In terms of tableaux, this involves synchronising the choices of possible worlds made during the reduction of the modal (sub)formulae that contain these atomic formulae as subformulae; or, in terms of positions, the reduction of the ν- and π-type positions that dominate the positions of the connection in the formula tree.

The following definitions are introduced for a given (indexed) formula tree for a given (indexed) formula X^{μ}.

Let T_M denote the union of $\mathcal{V}_0$ and Π_0. We associate a sequence of positions called a *prefix*, denoted $\mathrm{pre}(u)$, with each position u of the formula tree as follows: if $u_1 < u_2 < \cdots < u_n \leq u$, $1 \leq n$, are those T_M-elements that dominate u in the formula tree, then

$$\mathrm{pre}(u) = \begin{cases} (u_1\, u_2 \cdots u_n), & \text{K, K4, D, D4, T, S4;} \\ (u_n), & \text{S5.} \end{cases}$$

The prefix of a position encodes its modal context within the formula tree. We shall use p, q to denote prefixes.

For example, the prefix of $a_6^{(1)}$ is $\left(a_0 a_2 a_3 a_6^{(1)}\right)$ while the prefix of $a_{12}^{(1)}$ is $\left(a_0 a_{10}^{(1)}\right)$.

We can place various conditions on a binary relation $R_0 \subseteq T_M^{\star} \times T_M^{\star}$ as shown in Table 3. Such a relation is an *$\mathcal{L}$-accessibility relation* provided it satisfies the properties associated with $\mathcal{L}$ in Table 4.

Remark. These definitions are adapted from Fitting [Fit72,Fit83]. Each prefix "names" a possible world. Since the positions of the (indexed) formula tree correspond to signed subformulae of X^{μ}, a position taken together with its prefix corresponds to his notion of a *prefixed* signed (sub)formula. The prefix identifies the world in which the subformula is taken to be true or false depending on its sign. Binary relations on prefixes are thus used to represent the properties of the accessibility relation for a given logic. ■

Property	Condition: For $p, q \in T_M^*$
general	$p\ R_0\ pq$, $\|q\| = 1$
reflexive	$p\ R_0\ p$
transitive	$p\ R_0\ pq$, $\|q\| \geq 1$

Table 3: Prefix conditions.

$\mathcal{L}$	Properties of R_0
K, D	general
T	general, reflexive
K4, D4	general, transitive
S4	general, reflexive, transitive
S5	every prefix accessible from every other prefix

Table 4: Accessibility on prefixes.

We have indicated that the two positions that constitute a connection must be interpreted as inhabiting the same possible world; *i.e.,* have the same prefix. We ensure this by building a *modal substitution* σ_M under which the prefixes of the positions are identical. The discussion below motivates the ensuing definitions.

Consider a ν-type position u with prefix p. The semantic clause for the subformula rooted at u allows us to conclude that the subformula rooted at the child of u, say v, has the same truth value (sign) in *any* world accessible from the world denoted by p. By definition, the prefix of v is (pv) since v is of ν_0-type. Tables 3 and 4 give us the conditions under which a prefix can be considered to be accessible from p.

Take D4 for example. Any prefix of which p is a proper initial subsequence will be accessible from p. Consequently, if we consider v to be a "variable" and allow it to be instantiated under some mapping $\sigma_M: \mathcal{V}_0 \rightarrow T_M^*$ to any non-empty sequence we can guarantee that the image of (pv) under (the homomorphic extension of) σ_M will be accessible from the image of p under (the homomorphic extension of) σ_M. In the case of S4, we allow v to be instantiated with any sequence including the empty sequence to reflect the reflexivity of the S4 accessibility relation. For S5, since our notion of prefix is different, we need only consider unit sequences as possible instantiations for such "variables."

Now consider a π-type position u with prefix p. The semantic clause for the subformula rooted at u allows us to conclude that the subformula rooted at the child of u, say v, has the same truth value (sign) as u in *some* world accessible from the world denoted by p. Again, by definition, the prefix of v is (pv) since v is of a π_0-type. From the tables we can see that (pv) itself is accessible from p by virtue of the fact that accessibility relations on prefixes for all of the logics satisfy the general condition. Consequently we consider π_0-type positions as "constants" under the mappings σ_M introduced above. In the context of a tableau proof, the choice of this possible world must be arbitrary; *i.e.,* the prefix (pv) must be new to the tableau. The "constant" v can only be introduced in a prefix by the reduction of v's parent u, or by the reduction of a ν-type position introducing a "variable" (a ν_0-type position) whose image under σ_M contains v. To preserve soundness therefore, we must ensure that the former can occur before the latter.

A modal substitution $\sigma_M: \mathcal{V}_0 \rightarrow T_M^*$ induces an equivalence relation $\sim_M$ and a relation $\sqsubset_M$ on $T_M \times T_M$ as follows:

1. If $\sigma_M(u) = v$ for some v of ν_0-type, then $u \sim_M v$.
2. If $\sigma_M(u) = p$ and p is not a unit sequence consisting of a ν_0-type position, then for all $v \preceq p$,

$v \sqsubset_M u$; where $\preceq$ is the subsequence relation on $T_M^\star$.

3. If $v \sqsubset_M u$ and $u \sim_M u'$, then $v \sqsubset_M u'$.

A modal substitution σ_M is *$\mathcal{L}$-admissible* provided

1. σ_M respects $\mathcal{L}$-accessibility relations R_0, *i.e.*, for all $p, q \in T_M^\star$,

$$p\, R_0\, q \quad \text{implies} \quad \sigma_M^\#(p)\, R_0\, \sigma_M^\#(q)$$

where $\sigma_M^\#: T_M^\star \to T_M^\star$ is the homomorphic extension of σ_M to $T_M^\star$.

2. (K-logics only) $u \sim_M u'$ implies $v \sqsubset_M u$ (and hence $v \sqsubset_M u'$) for some position v.

3. $\lhd = (< \cup \sqsubset_M)^+$ is irreflexive, where $\sqsubset_M$ is the relation induced by σ_M described above.

The appropriate notion of complementarity for the propositional modal logics under consideration is as follows: (for an indexed formula X^μ) if σ_M is an $\mathcal{L}$-admissible modal substitution for X^μ a connection $\{u, v\}$ in X^μ is said to be *σ_M-complementary* iff

1. $\sigma_M^\#(\mathrm{pre}(u)) = \sigma_M^\#(\mathrm{pre}(v))$.

Remarks. The relation $v \sqsubset_M u$ should be interpreted as a prescription that "position v should be reduced before position u," in the sense of tableaux. The relation $\lhd$ is called the *reductio ordering.* Its irreflexivity ensures that we could construct an analytic tableau with X as root using the generic prefixes instantiated by σ_M, so that all of the restrictions on prefixes mentioned above are met. This method of representing the restrictions on traditional modal tableau rules is an adaptation of the method used by Bibel [Bib82a] for the classical quantifier rules.

Suitable mappings can be computed using variants on a string-unification algorithm. In all cases the set of most general unifiers is finite but not necessarily a singleton [Sie84]. For S5 the standard unification algorithm suffices. The admissibility check is an check for acyclicity if $\lhd$ is interpreted as a directed graph.

The extra condition for the K-logics is a translation into the current setting of Fitting's notion of a *used* prefix. Basically, since these logics are not idealizable we must ensure that each prefix (under σ_M) of a ν_0-type position (formula) has been introduced by the reduction of a π-type position (formula) beforehand. ■

We have proved the following theorem:

Theorem 2.5.3 *A propositional modal formula A is $\mathcal{L}$-valid iff there is a modal multiplicity μ_M, an $\mathcal{L}$-admissible modal substitution σ_M and a set of σ_M-complementary connections that spans the indexed formula $\langle A, 0\rangle^{\mu_M}$.*

The proof involves showing that starting from a tableau with $\langle A, 0\rangle$ at its root we can construct an atomically closed prefixed tableau by following the reduction ordering induced by the substitution, and prefixing each subformula with the image under the substitution of the prefix of its root position. The multiplicity indicates the number of times a given ν-type formula is reduced to form the tableau. Completeness involves showing that a suitable modal multiplicity μ_M can be constructed to form a modal Hintikka set from the set associated with any non-complementary atomic path (*i.e.*, unclosed branch) through $\langle A, 0\rangle^{\mu_M}$

Although we have used tableau systems to motivate the definition of the matrix systems, no tableau construction is actually performed in the use of such methods. The theorem above is utilised directly. (See Section 3.)

2.5.2 First-order modal systems.

Extending the propositional matrix systems presented above to first-order modal logics is straightforward. We consider both constant- and varying-domain versions.

For constant-domains, a pair of atomic formulae labelling the positions of a connection can be interpreted as complementary if we can find a *first-order* substitution σ_Q of parameters for individual variables that render the two atoms identical.

For varying-domains, the modalities and quantifiers interact. Universally quantified variables only range over those individuals that "exist" in the world denoted by the prefix of their quantifiers. Existential quantifiers express the existence of individuals only in the world denoted by their prefixes. Consequently, our first-order substitution σ_Q must respect the modal substitution σ_M.

Instead of introducing an explicit set of parameters we note that there is a 1-1 correspondence between Γ_0 and the set of distinct universally bound variables, and Δ_0 and the set of distinct existentially bound variables within the indexed formula. Consequently first-order substitutions are considered over these positions rather than individual variables. Notice that the position corresponding to the individual variable x quantified at a position u is the child of u in the formula tree.

More formally, let T_Q denote the set $\Gamma_0 \cup \Delta_0$. A first-order substitution is a mapping $\sigma_Q: \Gamma_0 \rightarrow T_Q$. For soundness, we must place restrictions on first-order substitutions to ensure that the positions representing parameters introduced for existentially bound variables (Δ_0) are indeed arbitrary. In terms of tableaux, we must ensure that such positions are introduced (by the reduction of their parent) before the introduction of any position representing a universally bound variable which receives the same parameter under the substitution σ_Q. The similarity between these restrictions on quantifier reductions and the restrictions on modal operator reductions is not accidental [Smu70].

A first-order substitution $\sigma_Q: \Gamma_0 \rightarrow T_Q$ induces an equivalence relation $\sim_Q$ and a relation $\sqsubset_Q$ on $T_Q \times T_Q$ as follows:

1. If $\sigma_Q(u) = v$ for some v of γ_0-type, then $u \sim_Q v$.
2. If $\sigma_Q(u) = v$ for some v of δ_0-type, then $v \sqsubset_Q u$.
3. If $v \sqsubset_Q u$ and $u \sim_Q u'$, then $v \sqsubset_Q u'$.

A *combined* substitution is a pair consisting of a modal substitution and a first-order substitution. A combined substitution $\langle \sigma_M, \sigma_Q \rangle$ is $\mathcal{L}$-admissible provided

1. σ_M respects $\mathcal{L}$-accessibility relations, as before.
2. (K-logics only) $u \sim_M u'$ implies $v \sqsubset_M u$ (and hence $v \sqsubset_M u'$) for some position v.
3. $\lhd = (< \cup \sqsubset_M \cup \sqsubset_Q)^+$ is irreflexive, where $\sqsubset_M$ and $\sqsubset_Q$ are the relations induced by σ_M and σ_Q respectively as described above and in § 2.5.1.

For constant-domains the appropriate notion of complementarity is as follows: (for an indexed formula X^μ) if σ is an $\mathcal{L}$-admissible combined substitution for X^μ, a connection $\{u, v\}$ in X^μ is *σ-complementary* iff

1. $\sigma_M^\#(\text{pre}(u)) = \sigma_M^\#(\text{pre}(v))$.
2. $\sigma_Q(\text{label}(u)) = \sigma_Q(\text{label}(v))$.

For varying domains complementarity is defined in the following way: (for an indexed formula X^μ) if σ is an $\mathcal{L}$-admissible combined substitution for X^μ, a connection $\{u, v\}$ in X^μ is *σ-complementary* iff

1. $\sigma_M^{\#}(\mathrm{pre}(u)) = \sigma_M^{\#}(\mathrm{pre}(v))$.
2. $\sigma_Q(\mathrm{label}(u)) = \sigma_Q(\mathrm{label}(v))$.
3. If $\sigma_Q(u') = v'$, then $\sigma_M^{\#}(\mathrm{pre}(u')) = \sigma_M^{\#}(\mathrm{pre}(v'))$.

Note the addition of the third clause by which the modal and first-order substitution interact.

Remark. We have blurred the distinction between an individual variable and the position that represents it in order to state the second condition. ■

Consequently we have:

Theorem 2.5.4 *A (first-order) modal formula A is $\mathcal{L}$-valid iff there is a multiplicity μ, an $\mathcal{L}$-admissible combined substitution σ and a set of σ-complementary connections that spans the indexed formula $\langle A, 0\rangle^{\mu}$.*

Once again we utilise tableau techniques to prove this theorem.

3 Proof search in the matrix systems.

The matrix systems presented above reduce the task of checking a modal formula for validity to one of path checking and complementarity tests. The path checking is performed by adding connections to a set and eliminating all those atomic paths that contain the new connection. If all atomic paths can be eliminated in this manner, the formula is valid. Complementarity tests are performed as each connection is added. Bibel [Bib82b,Bib82a] shows how some of the standard resolution search strategies can be utilised for this process. His results carry over to our modal systems without change.

Consider the example (indexed) formula of Figure 2. Ignoring the first-order features for the moment, the connection $\left\{ a_6^{(1)}, a_{12}^{(1)} \right\}$ gives rise to the problem of unifying the prefixes $\left(a_0 a_2 a_3 a_6^{(1)}\right)$ and $\left(a_0 \overline{a}_{10}^{(1)}\right)$, where we have overlined the ν_0-type ("variable") positions. We can immediately see that such a connection cannot be made (propositionally) complementary (condition 1) unless the accessibility relation of the logic is transitive. If this is the case, the (most general) unifier sends $\overline{a}_{10}^{(1)}$ to the sequence $a_2 a_3 a_6^{(1)}$.

The second connection $\left\{ a_8^{(1)}, a_{14}^{(1)} \right\}$ gives rise to the problem of unifying the prefixes $\left(a_0 a_2 a_3 \overline{a}_8^{(1)}\right)$ and $\left(a_0 \overline{a}_{10}^{(1)}\right)$. Since $\overline{a}_{10}^{(1)}$ has the value $a_2 a_3 a_6^{(1)}$ under the current modal substitution, we can make the two connections (propositionally) complementary if we send $\overline{a}_8^{(1)}$ to $a_6^{(1)}$.

Consider now the first-order features of our example formula. In addition to the modal substitution we must build a first-order substitution which unifies the labels of the connections.

For the first connection we must unify $Px^{(1)}$ with $Py^{(1)}$. This gives rise to the problem of unifying $\overline{a}_4^{(1)}$ with $a_{12}^{(1)}$. The most general unifier simply maps the former position to the latter. So far so good. Consider now the second connection. That gives rise to the problem of unifying $Qx^{(1)}$ with $Qz^{(1)}$, *i.e.*, $\overline{a}_4^{(1)}$ with $a_{14}^{(1)}$. Clearly we cannot build a consistent mapping for $\overline{a}_4^{(1)}$ which unifies both labels.

Due to the multiplicity of a_4 there is an alternative connection $\left\{ a_8^{(2)}, a_{14}^{(1)} \right\}$ which together with the first also forms a spanning set (§ 2.4). Propositionally, this connection gives us the problem of unifying $\left(a_0 a_2 a_3 \overline{a}_8^{(2)}\right)$ and $\left(a_0 \overline{a}_{10}^{(1)}\right)$ which is easily accomplished by mapping $\overline{a}_8^{(2)}$ to $a_6^{(1)}$ (recall that $a_{10}^{(1)}$ is already mapped to $a_2 a_3 a_6^{(1)}$). At the first-order level we must unify $Qx^{(2)}$ and $Qz^{(1)}$, *i.e.*, $\overline{a}_4^{(2)}$ with $a_{14}^{(1)}$ which can now be accomplished.

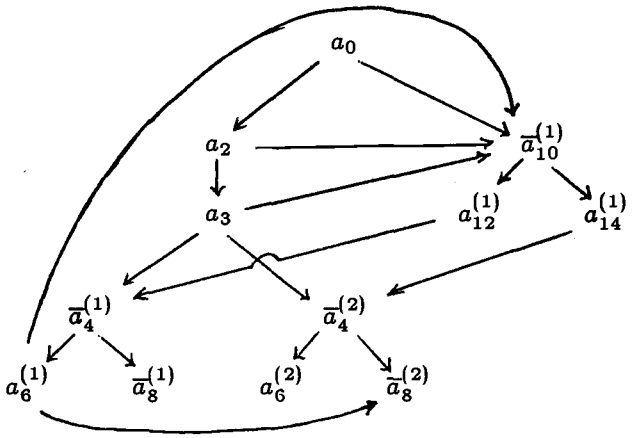


Figure 3: Reduction relation for connections 1 and 2′.

	Connection	M-prefix	σ_M	σ_Q	Q-prefix
1.	$a_6^{(1)}, a_{12}^{(1)}$	$a_0a_2a_3a_6^{(1)}, a_0\overline{a}_{10}^{(1)}$	$\overline{a}_{10}^{(1)} \rightarrow a_2a_3a_6^{(1)}$	$\overline{a}_4^{(1)} \rightarrow a_{12}^{(1)}$	$a_0a_2a_3, a_0\overline{a}_{10}^{(1)}$
2.	$a_8^{(1)}, a_{14}^{(1)}$	$a_0a_2a_3\overline{a}_8^{(1)}, a_0\overline{a}_{10}^{(1)}$	$\overline{a}_8^{(1)} \rightarrow a_6^{(1)}$	$\overline{a}_4^{(1)} \rightarrow a_{14}^{(1)}$	$a_0a_2a_3, a_0\overline{a}_{10}^{(1)}$
2′.	$a_8^{(2)}, a_{14}^{(1)}$	$a_0a_2a_3\overline{a}_8^{(2)}, a_0\overline{a}_{10}^{(1)}$	$\overline{a}_8^{(2)} \rightarrow a_6^{(1)}$	$\overline{a}_4^{(2)} \rightarrow a_{14}^{(1)}$	$a_0a_2a_3, a_0\overline{a}_{10}^{(1)}$
	propositional: $\sigma_M^{\#}(\text{pre}(u)) = \sigma_M^{\#}(\text{pre}(v)) \implies$				
	constant-domain: $\sigma_Q(\text{label}(u)) = \sigma_Q(\text{label}(v)) \implies$				
	varying-domain: for $\sigma_Q(u') = v'$, $\sigma_M^{\#}(\text{pre}(u')) = \sigma_M^{\#}(\text{pre}(v')) \implies$				

Table 5: Connections and unification problems.

The reduction ordering induced by these substitutions is shown as a graph in Figure 3. Notice that it is cyclic. It is easy to show that no increase in multiplicity can overcome this. Consequently we conclude that the formula is not valid in the first-order constant domain versions of the transitive logics.

We can also check the third condition to determine the status of the formula with respect to the varying-domain logics. Our first-order substitution mapped $\overline{a}_4^{(1)}$ to $a_{12}^{(1)}$ and $\overline{a}_4^{(2)}$ to $a_{14}^{(1)}$. The prefix of $\overline{a}_4^{(1)}$ is $(a_0a_2a_3)$ while the prefix of $a_{12}^{(1)}$ is $\left(a_0\overline{a}_{10}^{(1)}\right)$. Under the modal substitution this latter prefix becomes $\left(a_0a_2a_3a_6^{(1)}\right)$. Since these two prefixes cannot be unified the connections are not complementary in the varying-domain logics. (Notice that we do not even get as far as a cyclicity check in this case.) The prefixes and unifiers are summarised in Table 5.

The path checking process may be interpreted as constructing proof trees in a prefixed tableau or sequent based proof system where the prefixes contain "Skolem" variables and are interpreted as "Skolem" functions. The appropriate systems are similar in spirit to those of Jackson and Reichgelt [JR87]. This has been utilised in implementations to provide a human oriented interface to the search [WW87]. Note that we are concerned with an interface to the search itself rather than the presentation of an already constructed proof for which the techniques of [And80,Mil84] are applicable.

4 Related work

There are two main approaches for extending resolution techniques to modal logics. The first is to restrict the syntactic form of formulae, so that an appropriate modal clausal-form may be defined, and apply clausal resolution techniques (*eg.*, [Far83]). Bibel's comprehensive comparison of clausal resolution-based methods and his matrix method for first-order logic [Bib82b] suffices to demonstrate the advantages of proof search based on the matrix approach for modal logics presented above.

The second approach is to restrict the application of the resolution rule to modal contexts in which it is sound. In semantic terms this means utilising resolution within each possible world. Inference across possible worlds is performed by another mechanism. Abadi and Manna's systems [AM86a,AM86b], based on non-clausal resolution [MW80,Mur82], form perhaps the most comprehensive extension of resolution techniques to modal logics along these lines. The mechanism they employ to manage modalities are Hilbert-style deduction rules which are used to conjoin new formulae. For example, the modal deduction rules for S5 are:

$$\begin{array}{llll} \text{M1}: & \Box A, \Diamond B \mapsto \Diamond(\Box A \wedge B) & \text{M3}: & \Box A \mapsto A \\ \text{M2}: & \Diamond A, \Diamond B \mapsto \Diamond(\Diamond A \wedge B) & \text{M4}: & A \mapsto \Diamond A. \end{array}$$

While hand proofs using these systems can be short, the search spaces they generate are quite redundant due to the *connective*-based rules for manipulating modalities. Combinations of M3 and M4 must be aplied to facilitate the application of M1 and M2. Only when complementary subformulae are moved into the same modal context in this manner can the resolution rule be applied. Moreover, since the systems are generative, rules remain applicable to old formulae throughout the proof. This should be compared with our *connection*-based approach and the *calculations* used to establish validity illustrated in the previous section. In the example there, the propositional structure of the formula defined the space to be searched (four possible connections). The modal operators were dealt with using a unification algorithm.

Konolige's systems [Kon86] are based on tableau systems (one tableau for each possible world). Ordinary resolution is utilised within each tableau and a version of Stickel's Theory-resolution [Sti85a] used to manipulate modalities by creating new tableaux. Search is complicated by the need to choose suitable sets of formulae to form these new tableaux. The use of theory resolution is not effective, in the sense that an arbitrary amount of search must be performed to determine that the generation of a given resolvent is indeed sound. Konolige proposes the use of multiple refutation procedures to overcome these problems.

Acknowledgements

This research was supported in part by SERC/Alvey grants GR/D/44874 and GR/D/44270.

References

[AM86a] M. Abadi and Z. Manna. Modal theorem proving. In J.H. Siekmann, editor, *8th International Conference on Automated Deduction*, pages 172–189, July 1986. Lecture Notes in Computer Science, Volume 230, Springer Verlag.

[AM86b] M. Abadi and Z. Manna. A timely resolution. In *Proceedings of Symposium on Logic in Computer Science*, pages 176–186, June 1986.

[And80] P.B. Andrews. Transforming matings into natural deduction proofs. In W. Bibel and R. Kowalski, editors, *5th International Conference on Automated Deduction*, pages 281–292, 1980. Lecture Notes in Computer Science, Volume 87, Springer Verlag.

[And81] P.B. Andrews. Theorem-proving via general matings. *Journal of the Association for Computing Machinery*, 28(2):193–214, April 1981.

[Bib81] W. Bibel. On matrices with connections. *Journal of the Association for Computing Machinery*, 28(4):633–645, October 1981.

[Bib82a] W. Bibel. *Automated Theorem Proving.* Friedr. Vieweg & Sohn, Braunschweig, 1982.

[Bib82b] W. Bibel. A comparative study of several proof procedures. *Artificial Intelligence*, 18:269–293, 1982.

[Bib82c] W. Bibel. Computationally improved versions of Herbrand's Theorem. In J. Stern, editor, *Proceedings of the Herbrand Symposium, Logic Colloquium '81*, pages 11–28, North-Holland Publishing Co., 1982.

[BM72] R.S. Boyer and J.S. Moore. The sharing of structure in theorem-proving programs. In B. Meltzer and D. Michie, editors, *Machine Intelligence 7*, pages 101–116, Edinburgh University Press, 1972.

[Far83] L. Fariñas-del-Cerro. Temporal reasoning and termination of programs. In S. Amarel, editor, *8th International Joint Conference on Artificial Intelligence*, pages 926–929, 1983.

[Fit72] M.C. Fitting. Tableau methods of proof for modal logics. *Notre Dame Journal of Formal Logic*, XIII:237–247, 1972.

[Fit83] M.C. Fitting. *Proof methods for modal and intuitionistic logics.* Volume 169 of *Synthese library*, D. Reidel, Dordrecht, Holland, 1983.

[HM84] J.Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. In *3rd ACM Conference on the Principles of Distributed Computing*, pages 50–61, 1984.

[HM85] J.Y. Halpern and Y. Moses. A guide to the modal logics of knowledge and belief:preliminary draft. In *9th International Joint Conference on Artificial Intelligence*, pages 479–490, 1985.

[JR87] P. Jackson and H. Reichgelt. A general proof method for first-order modal logic. Submitted to IJCAI-87, 1987.

[Kan57] S. Kanger. *Provability in logic.* Volume 1 of *Stockholm Studies in Philosophy*, Almqvist and Wiksell, Stockholm, 1957.

[Kon84] K. Konolige. *A Deduction Model of Belief and its Logics.* PhD thesis, Stanford University, 1984.

[Kon86] K. Konolige. Resolution and quantified epistemic logics. In J.H. Siekmann, editor, *8th International Conference on Automated Deduction*, pages 199–208, July 1986. Lecture Notes in Computer Science, Volume 230, Springer Verlag.

[Mil84] D.A. Miller. Expansion tree proofs and their conversion to natural deduction proofs. In R.E. Shostak, editor, *7th International Conference on Automated Deduction*, pages 375–393, May 1984. Lecture Notes in Computer Science, Volume 170, Springer Verlag.

[Moo80] R.C. Moore. *Reasoning about knowledge and action.* Technical Note 191, SRI International, Menlo Park, Ca., 1980.

[Mur82] N.V. Murray. Completely non-clausal theorem proving. *Artificial Intelligence*, 18:67–85, 1982.

[MW80] Z. Manna and R. Waldinger. A deductive approach to program synthesis. *ACM Transactions on Programming Languages and Systems*, 2(1):90–121, 1980.

[Nis83] H. Nishimura. Hauptsatz for higher-order modal logic. *Journal of Symbolic Logic*, 48(3):744–751, September 1983.

[Pne77] A. Pneuli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*, pages 46–57, 1977.

[Pra60] D. Prawitz. An improved proof procedure. *Theoria*, 26:102–139, 1960.

[Sie84] J.H. Siekmann. Universal unification. In Shostak R.E., editor, *7th International Conference on Automated Deduction*, pages 1–42, May 1984. Lecture Notes in Computer Science, Volume 170, Springer Verlag.

[Smu68] R.M. Smullyan. *First-Order Logic*. Volume 43 of *Ergebnisse der Mathematik*, Springer-Verlag, Berlin, 1968.

[Smu70] R.M. Smullyan. Abstract quantification theory. In J. Myhill and R.E. Vesley, editors, *Intuitionism and Proof Theory*, pages 79–91, North Holland, Amsterdam, 1970.

[Sti85a] M.E. Stickel. Automated deduction by theory resolution. *Journal of Automated Reasoning*, 1:333–355, 1985.

[Sti85b] C. Stirling. *Modal logics for communicating systems*. Technical Report CSR-193-85, Dept. of Computer Science, Edinburgh University, 1985.

[Wal86] L.A. Wallen. Generating connection calculi from tableau- and sequent-based proof systems. In A.G. Cohn and J.R. Thomas, editors, *Artificial Intelligence and its Applications*, pages 35–50, Wiley & Sons Ltd., 1986.

[WW87] L.A. Wallen and G.V. Wilson. A computationally efficient proof system for S5 modal logic. In J. Hallam and C. Mellish, editors, *Advances in Artificial Intelligence*, John Wiley & Sons, 1987. Proceedings of AISB87, Edinburgh, Scotland, April 1987.