Automated Theorem-Proving in Non-Classical Logics

This dissertation contains original research carried out either by me independently or in collaboration, jointly or separately, with either Dr R.K. Meyer or Dr M.A. McRobbie. The parts of this thesis which are based on this joint research are clearly indicated in the text.

Paul Brian Thistlewaite

Abstract

This dissertation contains original research carried out either by me independently or in collaboration, jointly or separately, with either Dr R.K. Meyer or Dr M.A. McRobbie. The parts of this thesis which are based on this joint research are clearly indicated in the text.

Paul B. Thistlewaite
Abstract

The topic of this dissertation lies in the intersection of logic and computer science, and rests firmly in that area of artificial intelligence (AI) known as automated theorem-proving (ATP). Our principal concern is with the design and implementation of theorem-proving programs for a range of non-classical logics, and especially for relevant family of non-classical logics detailed in [Anderson and Belnap 75].

In Chapter 1 we discuss the history of and motivations for non-classical theorem-proving, concentrating on the uses within logic and AI of ATP systems based on relevant logic. In Chapters 2, 3 and 4 we develop automated theorem-proving techniques for relevant logic culminating in the program KRIPKE for deciding theoremhood in the relevant logic LR. Chapter 2 is concerned with replacing the proof-theoretical formulation of LR due to Kripke and Meyer with one more attuned to the requirements of automation. Chapter 3 advances the fundamentals of using algebraic models of a logic to prune the often immense search spaces generated during automated theorem proving in relevant logics. This technique is generalizable to theorem-proving in any logic that has an algebraic semantics. In Chapter 4 we discuss some of the strategic and extralogical considerations involved in making KRIPKE a viable ATP system.
# Table of Contents

1. Non-Classical Logic and Automated Theorem-Proving
   1.1. History of Non-Classical Logic in AI and ATP ... 2
   1.2. Non-Classical Approaches to AI ... 4
   1.3. Non-Classical Proof Procedures for Non-Classical Logics ... 16

2. Theorem-Proving for the Relevant Logic LR
   2.1. The Logic LR
      2.1.1. Syntactic Preliminaries and Axiomatization ... 25
      2.1.2. Facts about LR ... 29
      2.1.3. Proof Theory ... 33
      2.1.4. Decidability: formulation Lk ... 39
      2.1.5. Computational Complexity of Lk ... 51
   2.2. Minimal Proof Theory for LR
      2.2.1. Formulation L2: normal-forms and other equivalences ... 55
      2.2.2. Formulation L3: no contraction on \(+\) or \(\&\). ... 59
      2.2.3. Formulation L4: invertible rules and normalized proofs. ... 63
      2.2.4. Formulation L5: contraction confined; decidability. ... 67
      2.2.5. Computational Complexity Revisited. ... 76
   2.3. Properties of Provable Multisets
      2.3.1. Normal-Forms Revisited. ... 80
      2.3.2. Derived Axioms and the \(K\) Rule ... 83
      2.3.3. Positive/Negative Parts and the Rule-of-2 ... 87

3. Algebraic Models
   3.1. LR Models and the Matrix Property ... 93
   3.2. Measures of Matrix Strength
      3.2.1. Relative Measures of Strength ... 99
      3.2.2. Rigid Measures of Strength ... 102
   3.3. Testing for the Matrix Property ... 105

4. Proof Theory to Theorem Prover
   4.1. The Basic Algorithm
      4.1.1. The Preferred Strategy ... 126
      4.1.2. Adopting the Right Attitude ... 127
   4.2. Bits and Pieces
      4.2.1. Data Structures - or, Roman Numerals and Gödel Numbers are fine, but...
4.2.2. Applying Rules 149
4.2.3. Checking for Local and Global Properties 150
4.2.4. Program Correctness 154

Appendix A. Runtime Statistics for KRIPKE 157
Appendix B. Performance on Associative Formulas 170
List of Figures

Figure 2-1: The Weakening Rule K 38
Figure 2-2: The Proof of A4 in L1 38
Figure 2-3: Cut Rule for L1 39
Figure 2-4: Initial Segment of a Proof Search Tree 41
Figure 2-5: Examples of the Curry Property 45
Figure 2-6: AND/OR Graphs and the Curry Property 46
Figure 2-7: Dunn’s Picture of the Kripke Theorem 50
Figure 2-8: The Ko Rule 85
Figure 3-1: Matrices: CHAIN5 and CRYSTAL 98
Figure 3-2: Matrix 3 104
Figure 3-3: Matrix DIAMOND 108
Figure 4-1: AND-wise Failure 132
Figure 4-2: Static Costs vs. Dynamic Costs 141
List of Tables

Table 1-1: A Few Challenging Formulas 24
Table 2-1: Axioms and Rules for LR 26
Table 2-2: Formulation L1 37
Table 2-3: Formulation Lk 43
Table 2-4: Contraction Rules in L3 59
Table 2-5: The Po' and Pv' Rules of L4 67
Table 2-6: Formulation L5 68
Table 2-7: Premiss Pairs for Po''/Pko 79
Table 3-1: Sample of St1 versus 81 Measures 115
Table 3-2: Numbers of Models in n-Variables for Select $S$ 123
Table 3-3: Numbers of Non-Redundant Models for Select n and $S$ 123
Table 3-4: Percentages of Redundant Models for Select n and $S$ 123
Acknowledgements

My greatest debt is to my supervisors and friends Dr R.K. Meyer and Dr M.A. McRobbie. They have been my strongest source of influence, advice and encouragement.

My initial interest in the project of automating decision procedures for relevant logics I owe to Michael McRobbie, who proposed the project as an open problem in [McRobbie 79]. His enthusiasm and vitality in all matters logical is unbounded, and his interest and involvement in my work was no exception. Nor have I forgotten the favours, both professional and personal, that he has done for me. I am especially grateful for his having read and commented on all sections and aspects of this thesis.

Being able to study with Bob Meyer was indeed good fortune. The precision and subtlety of his thought, and the armoury of philosophical perspectives and formal strategies at his command, are formidable indeed, and I thank him for endeavouring to share them with me. I am especially grateful for the time he devoted to checking the formal proofs of this thesis, and in particular those proofs contained in Chapters 2 and 3.

I am also greatly indebted to Professor Alistair Urquhart (University of Toronto) and Professor J.M. Dunn (Indiana University) for further helpful conversations on the subject matter of Chapters 2 and 3.

My thanks go also to past and present members of the Australian National University Logic Group, especially Dr E.P. Martin (University of Melbourne) to...
whom I owe my initial interest in revolutionary logics and their freedom fighting cousins, Dr J.K. Slaney (University of Edinburgh) for leaving a trail of algebras behind him, Dr S. Giambrone (University of Southwestern Louisiana), Dr C. Mortensen (University of Adelaide), Dr R. Routley and Mr I. Urbas. My thanks also go to Mr D.S. Mannison (University of Queensland), and to Dr T.J. Richards (La Trobe University).

Finally, I wish to acknowledge the fact that this dissertation exists because of (and in part, despite) the assistance of several machines, and that any errors or omissions are of course ultimately due to them. The thesis itself was prepared using Unilogic’s Scribe package, and printed on a Canon Laserbeam Printer owned by the Department of Computer Science at the Australian National University. I should like to thank Dr R. Stanton, Head of that Department, for having made it available to me. The design and development work on KRIPKE was done primarily on DEC-10 and Vax 11/780 equipment, and I wish to thank the appropriate bodies at La Trobe University, the University of Melbourne and the Australian National University for access to this equipment.
Preface

As the research in this dissertation has drawn on several disciplines, including logic, computer science and mathematics, maintaining fidelity to the notational preferences of each discipline has not been possible, although the more central conventions have been preserved. By and large, our conventions will be those of some principal work in the area (e.g. [Anderson and Belnap 75] for relevant logic), and as the need arises we will explicitly cite the conventions of various works as the default for that section. We will warn the reader of any unavoidable and significant departures from this practice.

Generally, we shall adopt the following conventions. Logics or proof-theoretical formulations of a logic shall be named using bold faced characters - e.g. LR. Programming languages or particular programs will have names in small capitals - e.g. PASCAL. Function names will be lowercase italics - e.g. \( \text{crd}(A) \). Formal structures such as algebras will be named with uppercase italics - e.g. \( LATTICE \). We will let the symbols \( \leq, \geq, <, >, \in, \cup, \cap \) etc. retain their usual meanings, and we will let \( A, B, ..., Z \) range over sets. Occasionally we will define sets using set abstraction. Most things referred to may be subscripted or superscripted or both, and where context makes the meaning clear, subscripts or superscripts may be dropped. We note that our language for formulas is introduced in Chapter 2, algebraic terminology is given in Chapter 3, and our language for expressing computer algorithms is given in Chapter 4.
Chapter 1
Non-Classical Logic and Automated Theorem-Proving

The topic of this dissertation lies in the intersection of logic and computer science, and rests firmly in that area of artificial intelligence (AI) known as automated theorem-proving (ATP). Our principal concern is with the design and implementation of theorem-proving programs for a range of non-classical logics, and especially for the relevant or relevance family of non-classical logics detailed in [Anderson and Belnap 75].

In this chapter we discuss some of the history of and motivations for non-classical theorem-proving, concentrating on the uses within logic and AI of ATP systems based on relevant logic. In Chapters 2, 3 and 4 we develop automated theorem-proving techniques for relevant logic, culminating in the program KRIPKE for deciding theoremhood in the relevant logic LR. Chapter 2 is concerned with replacing the proof-theoretical formulation of LR due to Kripke and Meyer with one more attuned to the requirements of automation. Chapter 3 advances the fundamentals of using algebraic models of a logic to prune the often immense search spaces generated during automated theorem-proving in relevant logics. This technique is generalizable to theorem-proving in any logic that has an algebraic semantics. In Chapter 4 we discuss some of the strategic and extralogical considerations involved in making KRIPKE a viable ATP system.
1.1. History of Non-Classical Logic in AI and ATP

The centrality of ATP within the discipline of computer science, especially with some of the concerns of AI, is well known. [Nilsson 80], for example, mentions a range of connections between AI and ATP to support the claim that "... theorem proving is an extremely important topic in the study of AI methods" (p.5). We shall not examine these connections here. Similarly we will not repeat the general history of ATP nor detail the various successes or disappointments that the ATP enterprise has had during is relatively short life; see [Siekmann and Wrightson 83] for a good selection of references on these matters. Nor will we detail the methods, such as resolution, connection-graph, or natural deduction methods that are incorporated in or were significant in the development of classical automated theorem-provers, apart from occasionally noting when a particular classical method is inadmissible or unsound within a given non-classical theorem-proving environment.

For the purpose of displaying the importance of non-classical ATP, however, we will need to examine a little of the history of the general connection between logic and automated theorem-proving. Basic to ATP is the goal of programming machines to prove theorems. Theorems and proofs are among the objects principally of interest to logicians, and so it is not surprising to find that although ATP has had its champions, especially more recently, from within the discipline of computer science, a significant amount if not most of its foundational work relied heavily on antecedent work in logic. As this antecedent logical work was principally classical in nature - that is, it derived from the propositional and predicate calculi of [Whitehead and Russell 25] - ATP understandably premissed itself on classical notions of theoremhood, provability, and so on. With the popularization of non-classical logics and especially modal logics in the 1960's, the 1970's saw the direct use of non-classical logics in certain AI projects. [Morgan 76a] notes

"... the importance for artificial intelligence which is claimed for nonclassical (particularly modal) logics by various authors ..." (p.852)
and goes on to cite a considerable amount of literature to justify this claim. One of the central uses of modal logic has been to explicate the semantics of programming languages; see, for example, [Pratt 77]. Other non-classical logics, including 'fuzzy', temporal and relevant logics, have had their proponents, with the particular proposed application varying of course with the logic in question. Temporal logic has been suggested in [Pnueli 78] as a way of understanding concurrent processes, [Belnap 77] has suggested the use of relevant logics for question-answering systems, and [Shapiro and Wand 76] has proposed that relevant logics be used in intelligent database management systems.

Given the interest in non-classical logic within AI and the close relation between the problems of AI and the tools of ATP, one would expect a corresponding degree of interest in non-classical ATP. [Morgan 76a] (p.852), however, notes the disproportionately small amount of research invested into developing ATP systems for non-classical logics. Since Morgan's paper more ATP systems for modal and related logics have appeared, although the research into non-classical ATP that we are aware of is still far from extensive. For example, [Haspel 72] investigates modal and intuitionistic ATP systems, [Shimada 74] develops an ATP system for intuitionistic logic, [Morgan 76b] discusses resolution techniques for many-valued logics, [Pratt 77] provides algorithms for deciding dynamic logic, [Mogilevski and Ostrouhov 78] develops an ATP system for intuitionistic logic and the modal logic $S4$ using Smullyan analytic trees, [Mitani 78] looks at ATP strategies for intuitionism, [Farinas del Cerro 81] provides an ATP system for modal logic, [Wrightson 79] and [Fortenbacher, Schreck and Wrightson 81] investigate ATP systems for higher-order modal logic, [Cialdea 83] investigates resolution strategies for intuitionistic logic, and [Lauth 83] discusses resolution techniques for temporal logic.

Despite the apparent and growing interest in non-classical logic within AI, not

---

1 For our purposes, we omit work done on non-classical logics with finite characteristic matrices.
every researcher in ATP has been sympathetic to the use or development of non-classical ATP systems based on non-classical proof procedures. This resistance to non-classical ATP would seem to have two strands to it: some, like Bibel in [Bibel 82], intimate a rejection of non-classical logic itself and express total confidence in ATP's essentially classical origins, and others including Weyhrauch in [Weyhrauch 80] and Wos in [Wos 82], who seem to see the interest or importance of non-classical notions, nonetheless believe that non-classical ATP does not require the use of specialized non-classical proof procedures. We disagree with each of these views and the next two sections will examine each of them in turn.

1.2. Non-Classical Approaches to AI

Some rejections of non-classical logic are based on ignorance, and involve rash assessments of significant areas of the discipline of logic by people who lack the competence to do so. For example, Bibel's otherwise excellent book, [Bibel 82], is marred by the following passage:

"We have also left out the discussion of proof methods for non-classical logics, for instance for modal logic... Except for lacking competence this omission has its reason also in an uncertainty about the importance of such logics in the long run." [Bibel 82] (p.272)

Part of the evidence for this claim is the view to be discussed in the next section that, even if non-classical logics are important, we do not require non-classical proof methods to deal with them. The only other evidence Bibel cites for his claim is a line apparently contained in a private communication to him from J. McCarthy; namely

"modalities are important but modal logic may not be the right way to deal with them."

This would seem like an extremely cavalier rejection of what is in the eyes of many logicians of various persuasions a significant and important area of logic. To be fair to McCarthy's reported comment though, modal logic like most other non-classical logics initially received little attention and less than a lukewarm reception from
within the logic community of the day. Quine, for example, for various philosophical reasons believed (for quite a while at least) that there was no irreducible notion of necessity, and so resisted and promoted resistance to the idea that modal notions required special logics to account for them. This attitude to modal logic moderated with time, and with the advent of the Kripke semantics for modal logics it was seen that the modal connectives of different (formal) modal systems could take on plausible interpretations for modalities other than necessity and possibility. Not that the open metaphysical questions regarding modalities were thereby by-passed or closed, but modal logics became widely seen as plausible models of various modal concepts. As modal and other intensional notions became better understood and ontologically or logically more tolerable in certain circles, modal logics took on a certain respectability and acceptability within logic. By these remarks we do not want to suggest that certain modal notions are primary, or contra Quine, not eliminable in favour of or somehow logically reducible to more extensionally acceptable concepts or objects. The two-fold point is simply that, for a combination of reasons, modal logic is no longer unclean, and secondly even though modal notions might be reducible (contra the belief of the author) to classical ones, it is arguable whether it is necessary to effect the actual reduction, and it might be more natural or efficient to work in the theory of modality qua modal logic. (In much the same way that although Physics postulated the physical reducibility of any object to a collection of atoms, it tacitly admitted and promoted table-making by means other than nuclear fusion.) Gone are the days when card-carrying modal logicians were dragged before the Senate House Committee on Non-Classical Activities - we hope that McCarthy’s reported comment was not premissed on some (false) view that modal logic is still shunned within the discipline of logic.

By and large though, it seems that modal and other non-classical logics have had a better reception in AI than that which they initially received in logic. Possibly part of the reason for this is that the use of modal and related logics in AI has been seen somewhat instrumentally by many AI researchers - non-classical logics have
proved interesting because they appear to model certain types of situation or concept, of special interest to AI or computer science generally, in the seemingly most natural way. This instrumentalist attitude, intentional or otherwise, is to be commended; a formal system can answer to purposes other than the one it was born with and which originally motivated it, and using a non-classical logic need not carry with it any commitment to the *philosophical* motivations given by others for, say, preferring it to classical logic.

We do not mean to suggest that the AI community has been or should be insensitive to the philosophical motivations underpinning formal systems. Nor do we wish to suggest that the AI community is blind to the fact that non-classical logics often do not merely present *alternative* ways of modelling in more detail some aspect of a supposedly classical reality but rather will often challenge the very truth or reliability of the classical picture. Some of these challenges are no doubt of little immediate interest to AI, and especially ATP, and to researchers in these disciplines who do not have backgrounds in formal or philosophical logic the challenges and responses may occasional resemble the medieval scholastic debates concerning the number of angels that might dance on the head of a pin. For example, of what matter is it whether reality includes but the two truth values of classical logic, or the myriad truth values of many-valued logic, when in using the latter a researcher may perhaps not be concerned so much with Truth in all its Ontic and Metaphysical glory but rather, say, with modelling logical operations on information tagged with degrees of *epistemic* reliability?

But where a non-classical logic challenges the classical account of what *counts* as a deduction, valid inference, or proof, then the matter is of central importance to AI and ATP, for *proving* is the very object of the ATP exercise. If the claim of some non-classical logic is that classical logic gets the notion of implication wrong, then the corresponding ramification for classically-based ATP is that its methods may be faulty and unreliable. Indeed, modal, intuitionistic and relevant logics have all at one time or another been motivated as presenting accounts of implication
superior to the account given by classical logic. For example, the *strict* implication of the modal systems of [Lewis 18] were motivated by Lewis as presenting a better account of implication, in that strict implication avoided several of the paradoxical properties of the classical, *material* implication since the following theorems of classical logic are not theorems of these systems where implication is cashed out as strict implication (or for that matter, of relevant logics)

(i) \( p \rightarrow (q \rightarrow p) \)

(ii) \( \sim p \rightarrow (p \rightarrow q) \)

(iii) \( (p \rightarrow q) \lor (q \rightarrow p) \)

We may read (i) as saying that a true proposition is implied by anything, (ii) that a false proposition implies anything, and (iii) says that given any two unrelated propositions, at least one will imply the other. One response by classical logic, to Lewis' charge that the material account of implication produced paradoxical consequences, was of a "you too" kind. It was pointed out that the following paradoxes of material implication (amongst others) remained theorems of the Lewis systems when cashed out in terms of strict implication

(iv) \( (p \& \sim p) \rightarrow q \)

(v) \( p \rightarrow (q \vee \sim q) \)

where (iv) says that a contradiction implies anything, and (v) says that an excluded-middle is implied by anything. (It should also be noted that (i), (ii) and (iv) are theorems of intuitionistic logic as well.) Partly because of these paradoxes of strict implication, few of the logicians who were interested in investigating modal logics motivated this interest in terms of the absence of certain paradoxical implicational theses, and indeed, Lewis himself abandoned this motivation too.

Age has a way of betraying one's weaknesses, however, and so it is not surprising that the paradoxical theses, (i)-(v), concerning implication eventually came under renewed attack. This attack occurred in the 50's with the advent of the relevant family of logics. Relevant (or relevance) logics endeavour to confront the issues of implication including the paradoxes of both material and strict implication head on, and indeed none of (i)-(v) are theorems of any of the central relevant logics.
Broadly speaking, relevant logic regards a vertebrate theory of implication or valid reasoning as the cornerstone of a general theory of logic. This view has wide support within the logical community. For example, Quine claimed that

"The chief importance of logic lies in implication..." [Quine 51] (p.xvi)

Quine, of course, believes that implication is properly understood solely in terms of truth-functional relations. Preservation of truth maybe one aspect of good inferences, but having a diet consisting solely of truth-preservation makes for a poor theory of implication, as the paradoxes amply show. Relevant logics insist that the premisses of a valid implication be somehow relevant to what is implied.

[Anderson and Belnap 75] and [Routley, Meyer, Plumwood and Brady 84] present formal criteria for ‘relevance’, and we direct the reader to these texts for details of the relevant account of implication, and other philosophical and logical motivations for investigating relevant logics. Of course, logicians of the classical persuasion have tried to defend classical logic in the face of these criticisms by the proponents of relevant logic. [Routley, Meyer, Plumwood and Brady 84] contains a good though perhaps partisan account of these defences. We believe that classical logic is not without its strengths - as a model of state-descriptions and a logic of truth-preservation it is very successful. But inasmuch as the axioms of logic were not brought down from Mount Sinai etched immutably on stone tablets, but rather are postulated as theoretical best-fits of our logical intuitions, and whatever the virtues of material implication might be, we believe that the claim that material implication accommodates our intuitions regarding what follows from what stands severely indicted by the classical endorsement of paradoxes like (i)-(v). To use the material implication to link sentences in an implication relation is to risk irrelevance.

For these reasons, and because implication is so intimately concerned with central notions underlying AI, we believe that utilizing the perspectives and formal apparatus of non-classical and especially relevant logics is, contra Bibel and McCarthy, of considerable importance. Shortly, we will detail some of the areas where we see relevant logic as possibly making a considerable contribution to the
solution of certain central problems in AI. Before doing so, however, we would like to, by way of digression, mention two points concerning non-classical logic that might help dispell any misunderstandings about it.

The first point concerns the relation of classical to relevant logic, although it could be made with respect to classical and certain other non-classical logics besides relevant logic. Relevant logics need not be seen as antagonistic to classical perspectives, despite the fact that some of the preferred motivations for them, in for example [Anderson and Belnap 75] and [Routley, Meyer, Plumwood and Brady 84], are openly so. Classical logic, which following [Anderson and Belnap 75] we call TV, and the principal relevant logic R can each be formulated with the same set of primitive connectives - for negation, conjunction, disjunction and implication - and a Hilbert-style axiomatization given such that the axioms and rules for R are a proper subset of those given for TV (see Chapter 2.1 for details). Formulated this way, we see that (A) the theorems of R are a proper subset of the theorems of TV, and so R rejects some classical theorems (like the paradoxes of implication). Alternatively, as Meyer in [Meyer 78a] argues, classical logic can be seen as being contained, in a sense to be explained, in R and R can be viewed as an extension of classical logic. As is well-known, TV can be axiomatized simply by taking connectives for negation, and conjunction and/or disjunction, as primitive. Formulated this way, however, it is easy to show that (B) the theorems of TV are a proper subset of the theorems of R; see [Anderson and Belnap 75] (p.283) for details. There is no contradiction here between (A) and (B) - the truth of (A) relies on TV and R sharing the same primitive vocabulary, supposedly talking about one and the same concept of implication, whereas the truth of (B) relies on the primitive connectives of TV being a proper subset of those of R in which case TV is either seen as foregoing implicational formulas, or alternatively following the suggestion of [Meyer 78a] as defining an additional, material implication in the familiar way using negation and disjunction. We do not ourselves endorse either of

1It is this that permits conjunctive normal forms and other clausal equivalences of classical logic that in turn permit resolution techniques.
these perspectives, (A) or (B), over the other, but merely bring this choice to the attention of the reader: relevant logics can often been seen alternatively as restricting or supplementing classical logic.

Secondly, we note that some of the current, supposedly classical ATP systems do not, at least at the level of implementation, determine a preference for classical over non-classical logic. While some ATP systems are motivated from the perspective of classical logic - for example, Chapter 10 of [Clocksin and Mellish 81] motivates PROLOG this way - the implemented logic is occasionally, as in the case of PROLOG, Horn-clause logic which is a proper subsystem of not only of classical logic and but also of intuitionistic logic, and is thus properly non-classical. For further details on Horn-clause logic see [Galvin 70], and for details of the connection between PROLOG and intuitionistic logic see [Sakai and Miyachi 84]. In a sense, then, ATP systems like PROLOG have yet to determine their real logical parentage, and to motivate PROLOG from the perspective of classical logic is thus somewhat misleading.

Apart from the various logical advantages we have boasted of relevant logic, it would appear to commend itself to AI on other quite practical grounds also. One of the central uses of ATP is in the area of deductive database management, and its close cousin, question-answering systems. For some of the more basic connections between AI, deductive database systems and question-answering systems see [Nilsson 80], and for a more detailed discussion see [Gallaire and Minker 78].

The existence of logic highlights the fact that information is best stored not as a list of atomic, unrelated facts, but in such a way that the relations between pieces of information can be extracted and manipulated to produce new information. This process of revealing 'hidden' information from explicit data is exactly the process of logical deduction. Of course the reliability of derived information can only be as high as that from which it was derived, and it is no discredit to the deductive process that it cannot correct misdescriptions of the facts. The integrity of a database - its fidelity to the facts - is not and cannot be the concern of logic. Yet
this disclaimer is little consolation to the relational database designer wishing to use logic to unpack and reveal the relations between items in his database, for the brutal fact is that databases have a great propensity to become corrupted. Often, the explicit information they contain is at odds with the facts, and what is worse, often this information is at odds, explicitly or otherwise, with other information contained in the database. Obvious inconsistencies that display inconsistency syntactically can be detected at the time information is entered into the database, and the deductive process can be monitored to ensure that it does not add inconsistent information. But both of these attempts to secure a database from corruption are computationally expensive, probably ineffectual and logically unsatisfying.

Even when an inconsistency is detected, it is not always obvious which piece of information is the prime offender to be culled from the database. This is especially the case where an inconsistency only comes to light during the deductive process, and candidates for the offending item cannot be determined by simple syntactic means. For example, the following set of sentences is inconsistent:

(a) The President is in Moscow or the President is in New York.
(b) Anything in Moscow is in Russia.
(c) The President is not in New York.
(d) The President is not in Russia.

but this inconsistency would only be explicitly revealed in response, say, to a query as to whether the President was in New York, which would come out true. Even then, upon discovering that (a)-(d) are jointly inconsistent, it may not be an easy matter to discover which one (or more) of them should be discarded.

While the deductive process cannot be held responsible for misreported facts, it would be disconcerting if the presence of inconsistency were to totally disable the deductive process itself. Even if a classical ATP system resists explicit appeals to the paradoxes of material implication in order to prove some thesis, the paradoxes
are nonetheless theorems of classical logic, and may reside in the classical deductive closure of a set of sentences under classical modus ponens. If a database contains inconsistencies, and if the underlying logic does not 'instruct' the deductive process to prohibit the use of the paradoxes, then we are at a loss as to how using them can be systematically and reliably avoided.

It was essentially considerations like these that encouraged Belnap in [Belnap 77] and [Belnap 78] to propose the use of relevant logics in question-answering systems. Belnap's proposals are based on the use of the first-degree implicational fragment of \( R \) which has an elementary and uncomplicated decision procedure, but which does not permit the nesting of implications within sentences (although we do not regard this as a significant limitation). Automated deductions, based on database information, can in general be quite unlike classical deductions. For example, one of the defences of the classical account of good argumentation is that a sound argument is 'valid material implication plus the truth of premisses', but as Belnap rightly remarks especially of machine reasoning, we are often compelled by circumstance to reason with information that we've been told is true but which, whether we are aware of it or not, taken together is in fact inconsistent.

Belnap's work has received little attention from computer scientists,\(^1\) probably because [Belnap 77] and [Belnap 78] are not published within their mainstream literature. But the problems he raised concerning deduction in inconsistent databases have recently received quite a lot of attention, with the best known work being perhaps [McDermott and Doyle 80].\(^2\)

As [McDermott and Doyle 80] correctly points out, there would seem to be at least two levels of inconsistency that might arise in a database - one requiring 'routine revision' and the other requiring 'world-model re-organization'. Routine

\(^1\) [Shapiro and Wand 76] briefly describes some of the possibilities.

\(^2\) We note that [Doyle 78], on which the results of [McDermott and Doyle 80] are partially based, does cite [Belnap 77].
revision aims to repair inconsistencies that develop from particular universally
generalized statements having minor exceptions, which come to light as new
information is added to the database. They note that classical logic is deductively
helpless in the presence of any inconsistency, and add that
"... classical logic, by lumping all contradictions together, has
overlooked the possibility of handling the easy ones." [McDermott and
Doyle 80] (p.43)
They admit to having no solution to the general problem of inconsistency, but offer
a solution to the problem of routine revision. The proposal they make is for the use
of what they term a non-monotonic logic. The McDermott-Doyle description of
what constitutes a non-monotonic logic is, at the intuitive level, one in which
"...the introduction of new axioms can invalidate old theorems." (p.41)
In several places the terminology of [McDermott and Doyle 80] is somewhat non­
standard, but the intent is to supplement classical logic with a non-monotonic
deducibility relation. They note that classical logic simpliciter is monotonic, and
that their non-monotonic alternatives are non-classical.1 Unfortunately, by basing
their logics on classical logic, McDermott and Doyle explicitly include the
paradoxes of material implication (see clause 6(i), p.49) within their deductive
framework, and thus most of the problems concerning inconsistency in deductive
databases re-emerge. The logics they develop also have some internal counter-
intuitive properties; for example, they note (pp. 50-51) that their non-monotonic
logics fail to support a deduction theorem, which means of course that their new
non-monotonic deducibility relation is not directly connected with the implication
of their logics. Several of these weaknesses are discussed in [Gabbay 82], who

---

1 The fact that non-monotonic logics are non-classical has encouraged [Israel 80] to reject the
non-monotonic approach. Not that he fails to see the need for some (*pragmatic* or *epistemic*)
solution to database problems - it's just that he doesn't believe the (classical) logical account of
deducibility to be the problem, nor does he see it being the job of the logician to concern himself
with such questions. This task he hives off to the philosopher of science. Perhaps this is a verbal
issue, and perhaps it is just a question-begging defence of the classical conception of logic - what is
clear is that McDermott and Doyle and quite a number of other researchers in AI join with no less
a personage than Aristotle in believing that logic should be concerned with the principles of good
reasoning, and we join with McDermott and Doyle in seeing the problems they raise as being
questions about which are the principles of good reasoning.
proposes rather that intuitionistic logic would make a better candidate for the desired non-monotonic logic. He notes that intuitionistic logic, like classical logic, is not naturally non-monotonic, and the appropriate non-monotonic deducibility relations must be appended. He presents two non-monotonic logics based on intuitionistic logic but notes however that several features not desired of non-monotonic deducibility, related to the denial of excluded-middle in intuitionism, still emerge. We note again that intuitionistic logic also includes some of the paradoxes of material implication.

While it has been common knowledge within the area of relevant logic for some time, we bring the reader's attention to the fact that the relevant logic $\mathbf{R}$ is naturally non-monotonic. The preferred deducibility relation of $\mathbf{R}$ is already non-monotonic, and no new connectives need be introduced to get the desired effect. Moreover, $\mathbf{R}$ blocks the theoremhood of the paradoxes of material implication. We will not detail the properties of the relevant non-monotonic deducibility relation - for a nice account of relevant deducibility, see [Meyer and McRobbie 82], which also describes how $\mathbf{R}$ supports an appropriate version of the deduction theorem. We note however that the non-monotonicity of relevant deducibility is directly related to the use criterion that [Anderson and Belnap 75] provides, following [Church 51], for relevantly valid arguments.\textsuperscript{1} Even if, ultimately, the non-monotonic deducibility of relevant logic does not produce the 'clean' properties that [McDermott and Doyle 80] and [Gabbay 82] suggest are required for the purposes

\textsuperscript{1}There is also perhaps some connection between the M-connective of the McDermott and Doyle logics which makes their logics non-monotonic, and the $\ast$ operation of the Meyer-Routley *possible worlds* semantics described in [Routley and Meyer 73]. The Meyer-Routley semantics for $\mathbf{R}$ permits the modelling of non-trivial inconsistent theories, and essential to this is the unary operation $\ast$ which has the following intended interpretation: if $\vartheta$ is some world containing assumptions $A_1, \ldots, A_n$, then $\ast \vartheta$ is a complementary world in which $A_1, \ldots, A_n$ are not denied; i.e., $\ast \vartheta$ does not contain any of $\neg A_1, \ldots, \neg A_n$ (see [Routley and Meyer 73] p.202). We won't press the point here, but we recall Meyer during a seminar on default reasoning held several years ago muttering words to the effect "...its just star." It is likely to be more than that, but the two notions are clearly closely related.

Nor have we explored the relationship that may exist between the *co-tenability* interpretation of the relevant connective, $\circ$, for intensional conjunction provided by [Dunn 66] and [Dunn 75], and the M-connective.
of deductive database management, the utility of using relevant models of deduction is strongly indicated. To the extent that non-monotonicity in relevant logics is a *naturally* motivated and *integral* (rather than add-on) property of these logics, and given the fact that relevant deductive machinery is shielded against triviality in avoiding the paradoxes, relevant logics would seem like good testing grounds for non-monotonic hypotheses.

Handling inconsistent databases is just one of the areas in intelligent database management where employing relevant logics might help. Another problem concerning database management is, as [Plaisted 80] notes

"To solve problems in the presence of large knowledge bases, it is important to be able to decide which knowledge is relevant to the problem at hand." (p.79)

Indeed, Plaisted proposes several criteria for determining when pieces of information are relevant to the (possible) derivation of some conclusion, the main one being in terms of whether the literals (or propositional variables and constants) featuring in the information are *fully matched* (see p. 79). This ‘literal matching’ criterion of Plaisted’s is directly related to the *variable sharing* criterion that [Anderson and Belnap 75] propose for relevantly valid deductions. Given these close connections, the fact that [Anderson and Belnap 75] contains a wealth of formal and philosophical ideas regarding relevance, especially the relevance of premisses to conclusions, and the fact that relevant logics were initially intended to *be* logics of relevance, the utility of using them to partition large databases into (deductively) relevant parts is strongly indicated. We know of no work in computer science that has explicitly employed a relevant logic for this task, or which has even investigated the practical viability of doing so.
1.3. Non-Classical Proof Procedures for Non-Classical Logics

Granting then that non-classical logics are important, and of interest to AI, it is disconcerting that little research has been conducted into developing efficient non-classical proof procedures. Part of the reason for this is that as [Morgan 76a] rightly notes, many if not all non-classical (propositional) logics can be treated as higher-order (first-order) theories within classical logic. Indeed, some of the early research into non-classical ATP, such as that of [Haspel 72], employed exactly this approach. Morgan conjectures that,

"If automated theorem proving is to be developed as a practical tool either for the logician in the area of nonclassical logics or for the AI researcher wishing to use nonclassical logics, it seems that avoidance of higher order techniques is desirable." (p.852)

In contrast to this view some researchers, for example Weyhrauch in [Weyhrauch 80] and Bibel in [Bibel 82], have advocated that questions within non-classical logic, or non-classical approaches to questions in AI, be dealt with using higher-order theories and extant higher-order classical ATP systems, and have intimated that research into other non-classical ATP methods is a waste of time. Others, like Wos in [Wos 82] (p.17), explicitly advocate general purpose ATP systems and argue that implementing special purpose theorem-provers will not repay the effort. While this view is significantly milder than the first, we believe that it nonetheless implicitly contains a criticism of basing non-classical ATP on non-classical proof procedures.

To be blunt, we find the suggestion that non-classical ATP should be implemented using higher-order classical systems wrongheaded in the extreme. It is never contested that second-order ATP, for example, is considerably more difficult than first-order ATP, and conversely that specialized systems incorporating a great deal of local knowledge about the problem domain will often perform considerably better on problems in that domain than a general purpose problem solver. So if the logic of a given situation suggests or demands that, say, modal notions feature
centrally in some analysis of that situation, then it is foolishness not to have and to use the most suitable tool for providing that analysis. To suggest otherwise would be akin to recommending a four-wheel drive vehicle to a competitor in the Grand Prix on the grounds that a Jeep can take you anywhere that a Formula One can. Of what interest is it that Tool A has a wider range of possible applications than Tool B, if B not only fits one's purposes but fits better. On the contrary, the goal of ATP, and AI in general, should be to provide us eventually with the exact tool appropriate to the requirements of our task. And this approach is, after all, the basis of ATP's universal concern with eliminating redundancies, irrelevancies and bad choices from search spaces, and the prime motivation for seeking heuristic information of any kind. This is the stuff that efficiency is made of.

In response to the general-purpose versus special-purpose debate, the matter is clearly an empirical one and will often be decided one way or the other depending on how general or special the problem to be solved is. Our contribution to this debate will be to present several problems from logic that can be solved using a relevant ATP system employing proof procedures that we develop explicitly to handle relevant logics. We argue that these problems will be extremely difficult for extant general purpose systems to solve. We will return to this matter shortly, but firstly, we provide a general discussion of the techniques available for relevant ATP.

[Morgan 76a] outlines ATP methods for a range of non-classical logics including modal, many-valued, relevant and intuitionistic logics, but admits that these methods are not particularly suited to non-classical logics of the relevant or intuitionistic persuasion which, unlike modal logic, adopt non-classical notions of deduction. Despite Morgan's dislike of higher-order techniques, quoted above, his approach is very parasitic on such techniques. It does not surprise us then that the methods of [Morgan 76a] did not lead to "...a machine generated proof in this area which we would regard as particularly difficult or intricate.", a fact which Morgan freely acknowledges (p.862). None of the work contained in this dissertation...
borrows on Morgan's methods. To be fair, Morgan rightly acknowledges that "For certain specific logics, there may be more efficient methods than [his]" (p.861). We will totally confirm this in the case of relevant logics.

It is important to note that the difficulties involved at various levels of the classical ATP program set in sooner relevantly; for example, [Urquhart 77a] has recently proved that the principal relevant propositional logic, $R$, is undecidable, whereas undecidability does not arise classically until the first-order level. Inasmuch as first-order ideas are extensions of propositional ideas, a relevant ATP program requires a much more thorough understanding of relevant propositional ATP before embarking on ATP for higher-order relevant logics. One of the aims of this thesis to extend our understanding of relevant propositional theorem-proving, and to this end we develop the program KRIPTKE for automatically deciding theoremhood in a range of propositional relevant logics.

Although the work of Belnap, Shapiro and others did not attempt automated theorem-proving, and so the work of this thesis is the first on relevant proof techniques for relevant ATP, we will nonetheless endeavour to avoid the limitations of their work. In particular, the systems we will deal with will contain all formulas and theorems in the negation-implication vocabulary of $R$ rather than being confined to the first-degree implications, and will also treat formulas involving the so-called 'extensional' connectives, and the propositional constants of the principal system $R$. As we have noted $R$ is undecidable; however a large subsystem of $R$, called here $LR$, is decidable and meets our other desiderata. KRIPTKE, apart from being an automated theorem-prover for $LR$ and its proper subsystems and fragments which we introduce in the next chapter, has also been adapted to handle other logics, including the implication/negation fragment of the logic $E$ of [Anderson and Belnap 75], and the modalized relevant logic $NR_i$ of [Meyer 66].

Most of what we have to say about ATP for $LR$ applies to these other logics, and

---

1The name $NR_i$ derives from [McRobbie 79].
LR is the most intricate of the range of logics that KRIPKE can deal with. For these reasons we will confine our attention in what follows to ATP for LR.

Classical ATP is often based on resolution-style proof methods, which are unavailable for relevant logics. Resolution methods rely on having certain normal-forms, in particular conjunctive normal-forms, of the formulas of one's logic, with the admissibility of these normal-forms relying in turn on the provability within the logic of certain equivalences. As [Meyer 81] (p.278) notes, some of the equivalences that permit appropriate normal-forms in classical logic are not relevantly valid; for example, relevant implication fails to distribute over disjunction and so

$$p \rightarrow q \lor r \equiv (p \rightarrow q) \lor (p \rightarrow r)$$

is not a theorem of relevant logic. Hence KRIPKE will have to employ proof methods other than resolution.

Non-clausal techniques and approaches to ATP for classical logics, or fragments thereof, are not unknown however, and in some circles techniques other than resolution-based methods are preferred. There has been some research into ATP using Beth-Smulian analytic methods, for which see [Mogilevski and Ostrohov 78] and [Popplestone 67], and quite a comprehensive survey of other, mostly earlier, non-resolution theorem proving can be found in [Bledsoe 77]. Bledsoe expressed a preference for "natural" or "goal-directed" techniques, attributable to Gentzen, and predicted that these techniques would grow in popularity. This prediction would seem, in the light of [Murray 82], [Bibel and Schreiber 75], [Bibel 80] and [Bibel 82], to be fairly accurate. Gentzen-style proof-theoretic systems have been, as it happens, very important in the development of relevant logics since the late 1950's, and our development herein of ATP methods for LR will be based on Gentzen-style consecution formulations of the logic. The Gentzen formulations of LR that we will use will, of course, differ significantly from Gentzen-style consecution formulations of classical logic, especially in the structural rules permitted, and we caution the reader to watch for these differences. But the general approach should be familiar to sizeable sections of the classical ATP community.
We return now to the debate concerning general versus specialized ATP. We agree that, all too often in ATP perhaps, ideas come cheaply and do not pan out in terms of proofs of as-yet-unproven theorems, or improve on the modelling capabilities of existing ATP systems. To the extent that relevant ATP has not, at least up until this thesis, been extensively investigated, any such assessment of the idea of using specialized proof procedures for relevant ATP would have been unfair and certainly premature. With the advent of KRIPKE, we are in a position now to assess whether or not relevant ATP via specialized relevant proof procedures can solve problems of some complexity and interest which existing general ATP systems either cannot solve or have not solved as efficiently.

To anticipate matters somewhat, we are happy to announce that KRIPKE's worth lies not simply in its breaking ground, nor does it lie simply in being a handy desk-top logicians' helper (although neither are unimportant uses, as [Wos 82] rightly notes). But to the point, we mention briefly two particular problems - one from within ATP, and the other from within logic - that KRIPKE has been brought to bear fruitfully upon.

In [Ohlbach and Wrightson 83a] a solution is given to what is termed "Belnap's Problem", concerning the provability of the following formula in \( R \). Further details of the problem are given in [Ohlbach and Wrightson 83b].

\[
WF \quad (A \rightarrow (B \rightarrow B)) \rightarrow (A \rightarrow (A \rightarrow (B \rightarrow B)))
\]

Note that WF is also a theorem of \( LR \). The problem is somewhat misrepresented in that the provability of WF is of no special interest in relevant logic, nor is it difficult (even for a human) to prove. Its real interest lies, we believe, in the fact that it gives a way of comparing relevant ATP, based on proof techniques not specialized to relevant logics, with ATP using the specialized proof procedures of KRIPKE. [Ohlbach and Wrightson 83a] proved WF using (a modified version of) the Markgraf Karl Refutation Procedure; for details of which, see [Blasius et. al. 81]. The Markgraf Karl Refutation Procedure is a first-order classical ATP system, and the proof that WF was \( R \)-provable was effected by treating the propositional
relevant logic $\mathbf{R}$ as a first-order theory, using the Meyer-Routley semantics given in [Routley and Meyer 73] for $\mathbf{R}$ to define the theory. Given our earlier discussion of higher-order classical approaches to non-classical theorem proving, it was not surprising that the approach of [Ohlbach and Wrightson 83a] presented a proof of WF in around 10 minutes of CPU time, while KRIPKE can prove WF in about $1/10$th of a second. KRIPKE does not access any of its special global knowledge, such as that detailed in Section 2.3. about relevant provability, to effect this proof of WF; it follows fairly immediately on an unconstrained application of the appropriate Gentzen-style rules. This result clearly supports our claim that relevant ATP systems employing relevant proof procedures will perform more efficiently than generalized systems. We describe now a problem which was of immense logical interest to which KRIPKE was applied.

KRIPKE was the cornerstone of an attack on the decision problem for $\mathbf{R}$, details of which can be found in [McRobbie, Meyer and Thistlewaite 83]. This work was interrupted by Urquhart's brilliant proof in [Urquhart ??a] that $\mathbf{R}$ was in fact undecidable. [Urquhart ??b] discusses some of the ramifications of this result, including several unforeseen connections between relevant implication and projective geometry, and notes that $\mathbf{R}$ is perhaps the first independently motivated, undecidable propositional logic. The question as to whether $\mathbf{R}$ was decidable or not had been an open problem in logic for 25 years, and was of enormous logical difficulty. While Urquhart's proof utilized innovative associations of ideas, far beyond the conception of [McRobbie, Meyer and Thistlewaite 83], the two approaches shared the basic premiss that $\mathbf{R}$ was undecidable and that this undecidability could be related to the definability of an appropriately free associative connective within $\mathbf{R}$ to act like the desired semigroup operation in the manner of [Post 47].

Our version of this approach was to generate candidate definitions for such a connective, and prune the list of these candidates by showing that certain of them were not appropriately free - in particular, if a candidate definition defined a
connective which was provably associative in LR, then the fact that LR is
decidable would mean that the candidate definition could not be sufficiently free in
R. The second stage of our approach involved examining the remaining candidates
(if any) to see if one of them in fact defined an appropriately free associative
connective in R, using the techniques of [Meyer and Routley 73]. Some of the more
plausible candidate definitions are listed in Table 1-1. KRIPKE featured in the first
stage of our attack on the decision problem for R; that is, KRIPKE assisted in
establishing that each of our best candidate definitions, listed in Table 1-1, does in
fact define an associative connective in LR. At the time KRIPKE was rather
primitive, and failed to completely prove the associativity of any candidate; see
[McRobbie, Thistlewaite and Meyer 82] for details of this version of KRIPKE.
KRIPKE proved invaluable, however, in that it partitioned the defining formulas
into classes of formulas which are provably equivalent in LR; i.e. KRIPKE
automatically proved the equivalences (for details of these equivalences, see the
discussion of Impset in Appendix A). An interactive version of KRIPKE then acted
as a logicians’ helper when we established by hand, relying heavily on the tableau-
based techniques of [McRobbie and Belnap 79], that a representative of each such
class was provably associative in LR.

As we have mentioned, the version of KRIPKE used in the associativity project
was not all that sophisticated. In particular, none of the ideas of Chapter 2.2 of this
thesis, and few of the ideas in Chapters 2.3 and 3, had been developed at the time.
Although the question of the decidability of R has been solved, and our attack on it
has thus since lapsed, we note that KRIPKE can now prove at least one direction of
associativity for the connectives defined by most of the formulas in Table 1-1, and
has complete proofs of associativity for some of them. For exact details, see
Appendix B. The runtimes vary, and range up to about 20 minutes CPU time. The
best time is for the complete proof of the associativity of the connective defined by
No. 16 in the table, which KRIPKE provides in about 90 seconds. The proofs are, as
the reader would expect, quite long and so we have not reproduced them here;
copies can be obtained from the author. We commend the formulas in Table 1-1,
and the question as to their associativity in $\mathbf{R}$ or $\mathbf{LR}$, to the ATP community as a means of measuring the problem-solving strength of various ATP systems. On the basis of the respective performances of KRIPKE and the system of [Ohlbach and Wrightson 83a] at proving WF, we conjecture that extent general purpose first-order ATP systems will have considerable difficulty in proving the associativity of any of the connectives defined by the formulas in Table 1-1, and may even have difficulty proving the equivalences between defining formulas, mentioned above.
Table 1-1: A Few Challenging Formulas

Let the binary operations $\Phi_i$, $1 \leq i \leq 16$, be defined by the corresponding 16 formulas (for details of our language for formulas, see the opening sections of Chapter 2.1):

C1. $A \oplus (A \oplus B) \oplus (B \oplus A)$
C2. $A \oplus (A \oplus B)$
C3. $A \oplus (A \oplus B)$
C4. $A \oplus (A \oplus B)$
C5. $A \oplus (A \oplus B)$
C6. $A \oplus (A \oplus B)$
C7. $A \oplus (A \oplus B)$
C8. $A \oplus (A \oplus B)$
C9. $A \oplus (A \oplus B)$
C10. $A \oplus (A \oplus B)$
C11. $A \oplus (A \oplus B)$
C12. $A \oplus (A \oplus B)$
C13. $A \oplus (A \oplus B)$
C14. $A \oplus (A \oplus B)$
C15. $A \oplus (A \oplus B)$
C16. $(A \oplus (A \oplus B)) \oplus (B \oplus (B \oplus A))$

The question of whether these define associative connectives in $R$ or $LR$ then amounts to whether each of $(C \Phi_i D \Phi_i E) \Rightarrow ((C \Phi_i D) \Phi_i E)$, for $1 \leq i \leq 16$, is provable in $R$ or $LR$. For example, in the case of $i=16$, this amounts to whether the following formula is provable:

$$(((Cv(\neg C)) \circ (Dv \neg D)) v (((Cv(\neg C)) \circ (Dv \neg D)) v (((Cv(\neg C)) \circ (Dv \neg D))) \circ (Ev \neg E)) \rightarrow (((Cv(\neg C)) \circ ((Dv(Do \neg D))) \circ (Ev \neg E)) v (((Dv(Do \neg D))) \circ (Ev \neg E))) \text{ & }$$

$$(((Cv(\neg C)) \circ ((Dv(Do \neg D))) \circ (Ev \neg E)) v (((Dv(Do \neg D))) \circ (Ev \neg E))) \rightarrow$$

$$(((Cv(\neg C)) \circ (Dv \neg D)) v (((Cv(\neg C)) \circ (Dv \neg D)) v (((Cv(\neg C)) \circ (Dv \neg D)))) \circ (Ev \neg E)))$$
Chapter 2
Theorem-Proving for the Relevant Logic LR

In this chapter we firstly introduce the relevant logic LR, provide it with a Hilbert-style axiomatization, and collect some metalogical facts concerning it, including the proof due to Kripke and Meyer that LR is decidable. In Section 2.2 we explore alternative proof-theoretic formulations of LR with an eye to developing a computationally optimal proof-theoretic formulation of LR. A preferred proof-theoretic formulation of LR - a Gentzen-style consequence calculus for LR which we call L5 - is developed and shown to be a dramatic improvement, computationally, on all previous formulations of LR. We prove that L5 is both decidable, and sound and complete with respect to LR. In Section 2.3 we develop other logically motivated techniques for further improving the efficiency of KRIPKE.

2.1. The Logic LR

2.1.1. Syntactic Preliminaries and Axiomatization

Our propositional language PL is a structure consisting of a denumerable set of propositional variables, \{p,q,r,p_1,...\}, the propositional constants t to be interpreted as the conjunction of all theorems and T to be interpreted as the disjunction of all propositions (for which see [Anderson and Belnap 75] (p.342)), the unary connective ~ for negation, the following binary connectives, given in order of decreasing binding strength:

\begin{align*}
& & Conjunction \\
v  & & Disjunction
\end{align*}
Implication and formulas, with schematic variables for formulas being $A, B, C, D, E, A_1, ...$

The set of formulas $\mathcal{PL}_f$ is defined recursively as the smallest set such that

- if $A \in \{p, q, r, p_1, ...\} \cup \{t, T\}$ then $A \in \mathcal{PL}_f$
- if $A, B \in \mathcal{PL}_f$ then $\neg A, A \land B, A \lor B, A \rightarrow B \in \mathcal{PL}_f$

Our bracketing conventions are those of [Church 56] augmented where necessary by those of [Anderson and Belnap 75], and we will occasionally suppress brackets where the precedence of connectives makes the meaning clear. Formulas which are either propositional variables or constants or single negations of them will be called literals, and formulas involving at least one binary connective will be termed complex formulas. Our metalanguage will be ordinary mathematical English, and we will symbolize "formula $A$ is a theorem of the logic (or a formulation of the logic) $L$" as $\vdash L A$.

We now provide a Hilbert-style axiomatization of LR:

**Table 2-1: Axioms and Rules for LR**

### Implication Axioms

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>$A \rightarrow A$</td>
</tr>
<tr>
<td>A2</td>
<td>$A \rightarrow B \rightarrow C \rightarrow A \rightarrow C \rightarrow B$</td>
</tr>
<tr>
<td>A3</td>
<td>$A \rightarrow B \rightarrow B \rightarrow C \rightarrow A \rightarrow C$</td>
</tr>
<tr>
<td>A4</td>
<td>$(A \rightarrow A \rightarrow B) \rightarrow A \rightarrow B$</td>
</tr>
<tr>
<td>A5</td>
<td>$(A \rightarrow B \rightarrow C) \rightarrow B \rightarrow A \rightarrow C$</td>
</tr>
</tbody>
</table>
Implication/Negation Axioms

A6. \( \sim \sim A \rightarrow A \)
A7. \( A \rightarrow \sim A \rightarrow \sim A \)
A8. \( A \rightarrow \sim B \rightarrow B \rightarrow \sim A \)

Double Negation
Reduction
Contraposition

Conjunction/Disjunction Axioms

A9. \( A \& B \rightarrow A \)
A10. \( A \& B \rightarrow B \)
A11. \( (A \rightarrow B) \& (A \rightarrow C) \rightarrow A \rightarrow B \& C \)
A12. \( A \rightarrow A \lor B \)
A13. \( B \rightarrow A \lor B \)
A14. \( (A \rightarrow C) \& (B \rightarrow C) \rightarrow A \lor B \rightarrow C \)

Simplification
Simplification
&-Introduction
Addition
Addition
\lor-Introduction

Axioms for Constants

A15. \( (t \rightarrow A \rightarrow A) \& (A \rightarrow t \rightarrow A) \)
A16. \( A \rightarrow T \)

Rules

R1. If \( \vdash \neg A \) and \( \vdash \neg A \rightarrow B \) then \( \vdash \neg B \)
R2. If \( \vdash \neg A \) and \( \vdash \neg B \) then \( \vdash A \& B \)

Modus Ponens
Adjunction

Note that some of these axioms are redundant, but we will not be concerned with the question as to which axioms are independent. Note also that LR can be formulated without the propositional constants, but as [Meyer 77] points out, they have proved so theoretically useful in the development of relevant logics that their inclusion is desirable. Certainly, if we want to use our theorem-prover to discover facts about LR, their inclusion can only help.

We now define the propositional constants \( F \) and \( f \):

Deff: \( f =_{df} \sim t \)
DefF: $F = \text{df} \sim T$

and the binary connectives $\&$, $+$ and $\Rightarrow$:

Def$\&$: $A \& B = \text{df} \sim (A \to \sim B)$ Fusion
Def$+$: $A + B = \text{df} \sim A \to B$ Fission
Def$\Rightarrow$: $A \Rightarrow B = \text{df} (A \to B) \& (B \to A)$ Equivalence

Fusion and fission are intensional analogues of $\&$ and $\lor$ respectively. We note that we could have taken each of these symbols as primitive, supplying appropriate axioms for them in Table 2-1. For example, the following axioms would suffice for fusion:

$$A \to (B \to A \& B)$$

$$(A \to (B \to C)) \to ((A \& B) \to C)$$

For details of these alternative formulations, see [Anderson and Belnap 75] p.344.

From this axiomatization of $LR$ we obtain the principal relevant logic $R$ by simply adding the following axiom:

A17. $A \& (B \lor C) \to (A \& B) \lor (A \& C)$ Distribution

The implication/negation fragments of $R$ and $LR$ are the same, and we call the logic defined by the axioms and the rule governing these connectives - i.e. A1-A8 and R1 - the logic $R_i$, following [McRobbie 79]. From this axiomatization of $R$ we can obtain the classical propositional logic $TV$ by adding the (paradoxical) implicational axiom

A18. $A \to B \to A$

$LRQ$, the first-order extension of $LR$, can be axiomatized by taking the axiomatization for $RQ$ which is the first-order extension of $R$ provided in [Meyer, Dunn and Leblanc 74], and then dropping A17 and the confinement axiom

A19. $(\forall x)(A \lor B) \to (\forall x)A \lor B$ $x$ not free in $B$.

Propositional quantifiers can be added to $LR$, resulting in the logic $LRp$, along the lines of [Meyer 78b]. The argument of [Meyer 78c] that the implication/conjunction/universal-quantifier fragment of propositionally quantified $R$ is undecidable can be easily adapted to show $LRp$ undecidable. Little work has been
done in the area of LR-theories. [Meyer and Mortensen 82] have posed a few
questions concerning number theory based on LR, called LR#, which is just LRQ
together with Peano's postulates (or alternatively, just the theory R# of [Meyer
76, Meyer 75a, Meyer 75b] without A17). The amount of number theory that can
actually be done in LR# remains to be seen.

Finally, we note without proof that the following De Morgan equivalences are all
theorems of LR.

T1. \( \neg(A \lor B) \equiv (\neg A \land \neg B) \)
T2. \( \neg(A \land B) \equiv (\neg A \lor \neg B) \)
T3. \( \neg(A \lor B) \equiv (\neg A \land \neg B) \)
T4. \( \neg(A + B) \equiv (\neg A \land \neg B) \)

and that the connectives \& \lor \land \land + are each provably associative and
commutative in LR. The connectives \& and \lor differ from their intensional
counterparts \land \land + respectively in that whereas \& and \lor are idempotent in LR, \land
and + are not. These facts may be extracted from [Anderson and Belnap 75]
(p.396).

2.1.2. Facts about LR

To those familiar with the area of relevant logic the logic LR is no newcomer,
but the name is new. As we have seen in the previous subsection LR is just the
principal relevant logic R minus the distribution axiom. This fact prompted most
researchers to refer to the logic as R-distribution, or simply as R- as in [Meyer 66].
However R and LR share the fragment R_i, and more importantly R and LR are
each conservative extensions of R_i (for which see [Meyer 72]), and so LR agrees
totally with R concerning what might be termed the purely relevant insights into
implication. Nonetheless, to be fair to LR's critics, LR may not be the One True
Logic simply because it lacks what most of us would consider a reasonable principle
of logic, namely distribution itself. The unprovability of A17 in LR leads to the
omission of other popular though controversial principles of reasoning; in
particular, to the failure of disjunctive syllogism,
DS. \( A \& (\sim A \lor B) \to B \)

and even the failure of the rule-form of DS, namely \textit{modus ponens} for material implication or \( \gamma \):

\[
\gamma. \quad \text{If } \vdash A \text{ and } \vdash \sim A \lor B \text{ then } \vdash B
\]

It is well known that adding DS to relevant logics leads to collapse back to TV. [Meyer and Dunn 69] have shown that DS in rule-form, or \( \gamma \), is an \textit{admissible rule} in the full logic \( R \) - i.e. its inclusion as a primitive rule does not alter the stock of theorems of the original logic. However, [Anderson and Belnap 75] (p.298-299) note that in the absence of A17, \( \gamma \) fails, with the following counter-example being due to Meyer

\[
\vdash A \to A \text{ and } \vdash (A \to A) \lor ((A \to A) \& B) \lor \sim B
\]

but not \( \vdash ((A \to A) \& B) \lor \sim B \).

Thus, \( \gamma \) is not an admissible rule in LR. In the spirit of [Meyer, Giambrone and Brady 87] we consider this neither a good nor bad feature of LR. Nor do we mind if, in the spirit of [Belnap and Dunn 81], some people find this a \textit{good thing}. We suspect, however, that the absence of A17, DS and \( \gamma \) would not win LR too many votes in quite a few quarters. In particular, the inadmissibility of \( \gamma \) means that LR, unlike R, does not contain the tautologies of TV.

But the world is full of logics, each with their relative strengths and weaknesses, depending on one’s purposes, and all logics should receive their due. Indeed, the fact that LR lacks distribution led [Routley, Meyer, Plumwood and Brady 84] to see LR as a possible basis for a relevant Quantum Logic, which prompted the name Ortho-R, contracted to OR in [McRobbie 79]. Unfortunately, inspection of the axioms for LR reveals that there is nothing particularly ‘ortho’ about the logic at all. [Routley, Meyer, Plumwood and Brady 84] were not mistaken regarding the possibilities however, and indeed [Kron, Marie and Vujosevic 81] have recently proposed a candidate relevant Quantum Logic by adding the orthomodular law

\[
A \& (\sim A \lor (A \& B)) \to B
\]
to LR to get the logic they call $R^tQ$.\footnote{We think that $R^tQ$ with the addition of Peano's postulates and quantifiers may be of more interest than LR#; given the work of [Dunn 82] concerning the derivability of distribution in a number-theoretic form within the number-theoretic extensions of certain orthomodular logics.}

Despite some of the logical oddities of LR, we believe that the fact that LR contains $R_i$, shares the full propositional language $PL$ with $R$, and is the largest natural fragment of $R$ known to be decidable, makes it a good starting place for any investigation of the possibilities for relevant ATP. We now provide the reader with more background concerning LR, but before doing so, we justify our choice of name for the logic. We intend LR to be a contraction of the term Lattice-R, and an inspection of the axiomatization we have provided for LR will show that the conjunction and disjunction connectives have exactly the properties of the lattice operations of meet and join.

As will become clear from later sections of this thesis, we will not only be concerned that Kripke implement proof procedures for a decidable relevant logic, which LR is, but that these procedures be indeed decision procedures. Our principal reason for requiring this is to ensure that negative answers, of the form that formula A is not provable in LR, are at least possible. Unlike classical and modal logics, the connections between proof-theoretical and semantical perspectives of relevant logics are poorly understood, and there is no known way in the case of $R$ or LR of extracting from an attempted proof-theoretic proof of some unprovable formula A, information that would generate a semantical counter-model of A is there is one. Thus, the only way we know of for ensuring that negative answers are (in principle) possible is for our proof procedure to examine, if necessary, all possible proofs of a formula.

While on the subject of semantics, we note in passing that there has been some controversy in the literature as to whether the relational Kripke-style semantics provided by [Routley and Meyer 73] for the various relevant logics including $R$. 
indeed constitutes a ‘real’ semantics; see, for instance, [Copeland 79] and [Routley, Routley, Meyer and Martin 82]. We are temporarily spared this debate in the case of LR, in that the logic has resisted attempts to provide it with a useful Kripke-style semantics. The Meyer-Routley semantics for R, and in particular the interpretation clauses for conjunction and disjunction, straightforwardly verify all the conjunction and disjunction axioms for R, including A17. Moreover, these clauses seem natural for the connectives and so it is hard to motivate let alone devise modifications to the Routley-Meyer semantical picture which will exclude distribution and retain the rest. The Meyer-Routley picture is probably the wrong one for LR anyway, the point being that LR does not have a full-fledged disjunction, as evidenced by the very failure of axiom A17.

LR does have an algebraic semantics. Adopting an algebraic perspective has lead to numerous important insights into various relevant logics, and non-classical logics generally; for examples, see [Dunn 66], [Meyer and Routley 72], [Rasiowa 74], [Dunn 75] and [Slaney 80]. Chapter 3 will explore the insights into relevant ATP that arise by adopting an algebraic perspective. Immediately pertinent to our concerns here is an unpublished result of Meyer’s to the effect that although LR has no finite characteristic algebraic model or matrix, it does have the finite model property - i.e. any non-theorem of LR can be refuted in a finite model.\(^\text{1}\) Together with the fact that LR is finitely axiomatizable, a algebraic proof of the decidability of LR follows immediately. Meyer’s proof is via a translation of LR into the implication-conjunction fragment of R, which was shown to have the finite model property in [Meyer 73]. The algebraic decidability proof for LR involves an enumeration of algebraic models, and so is not a viable basis for KRIPKE.

\(^1\)This result of Meyer’s was presented to a seminar of the Logic Group of the Australian National University in 1983.
2.1.3. Proof Theory

In the next subsection we will detail the Kripke-Meyer proof-theoretic decision procedure for LR. Before doing so, we will sketch the major proof-theoretical formulations of LR. Counting styles, they number three:

• various natural deduction formulations, such as the Fitch-style natural deduction formulation which can be extracted from [Anderson and Belnap 75] (pp.346-348) by deleting the Dist rule for R.
• a proof-theoretically based tableau formulation developed in [McRobbie and Belnap 77].
• and Gentzen-style consecution calculi

Apart from noting that the Fitch-style natural deduction formulation sits better without the unmotivated rule required to prove distribution, we will leave it to one side. The tableau formulation of [McRobbie and Belnap 77] (see also [McRobbie and Belnap 79]) is in fact a refutation system, and while it has so far been of only limited theoretical usefulness, it is unquestionably the easiest of the three methods to use. The interesting results for LR have come mainly from various the Gentzen formulations of this logic. They either draw from or can be extracted directly from [Kripke 59, Belnap and Wallace 61, Belnap and Wallace 65, Meyer 66, Anderson and Belnap 75, McRobbie 79] and [McRobbie and Belnap 79].

A Gentzen-style consecution formulation of a logic takes objects of the form CN. \( \mu \vdash \nu \) as primitive, where \( \mu \) and \( \nu \) are collections (e.g. sets, sequences, etc.) of formulas, and \( \vdash \) is a primitive symbol whose intended interpretation is that the formulas on

---

1This point is noted by many; see [Read 80]. Proof-theoretically, relevant logics seem more natural without distribution, unlike the situation semantically, where excluding distribution becomes problematical.

2At least as far as people are concerned. The tableau formulation is not suited to ATP, because it resists the incorporation of many of the node-pruning devices used in KRIPKE.
the right of the ├ can be derived in the logic from those on the left. Objects like CN are called *consecutions*. A Gentzen formulation of a logic stipulates a privileged class of consecutions, called *axioms*, and postulates two sets of rules: *connective rules* for introducing complex formulas into consecutions, and *structural rules* for manipulating the number and order of formulas on either side of a consecution. The general motivational remarks of [Bibel 77] incline us to use a right-sided consecution formulation which, true to our enterprise, would provide *proofs* of LR theorems. A *right-sided Gentzen formulation* is one in which all objects take the form CN with the left-hand sequence, µ, null. The Gentzen formulations we present for LR will be right-sided, and will take ν as a *multiset* of formulas.

Throughout this thesis we will make substantial use of the concept of a *multiset*, which has had some currency in the computer science literature under the name of ‘bag’. Multisets are unordered collections of objects which can contain multiple occurrences of the same object. In the former respect they are exactly the same as sets, and in the latter respect they are exactly the same as sequences. They have been taken as datatypes in a number of programming languages - e.g. QLISP, for which see [Rieger, Rosenberg and Samet 79] - and they have been used in a number of theoretical studies in computer science and in logic - e.g. [Dershowitz and Manna 79] and [Meyer and McRobbie 82]. As [Hickman 80] notes, there is presently no standard notation for various multiset concepts, although this situation is considerably remedied by the recent work of [Meyer and McRobbie 82] and [Monro 83]. Our notation for multisets will be borrowed from there.¹

Given some arbitrary (finite) collection of objects we shall formally represent the multiset containing them by enclosing them in double square brackets. Thus

¹Monro distinguishes what he terms *multinumbers* from *multisets*, with our ‘multisets’ being his ‘multinumbers’ more or less. We believe that although Monro’s distinctions make good sense, his claim that his ‘multiset’ is "...the more set-like of the two" is not obvious - e.g. the power multinumber of a multinumber is more like that of a power set of a set than is Monro’s power multiset.
\[[A,A,A,B,C,C]\] is a multiset containing three occurrences of A, one of B and two of C. It is identical to the multiset \[[C,A,A,B,A,C]\], but not identical to the multiset \[[A,B,C,C]\]. We will let
\[\alpha, \beta, \gamma, \delta, \epsilon, \alpha_1, \ldots\]
be variables ranging over multisets, and we will let \[\square\] be the null multiset. Every multiset is associated in a natural way with a certain set - namely the set containing just the members of the multiset. We shall call the members of this set the *generators* of the multiset. Thus given a multiset \(\alpha\), we let \(g\) be a function such that \(g(\alpha)\) is the set containing exactly the generators of \(\alpha\); e.g. \(g([A,A,B,C]) = \{A,B,C\}\), \(g([\square]) = \{\}\).

**Definition 2.1:** We say that a formula \(A\) is a *member* of a multiset \(\alpha\), in symbols \(A \in \alpha\), iff \(A \in g(\alpha)\), and where \(g(\alpha) = g(\beta)\) we say that \(\alpha\) and \(\beta\) are *cognate*, written as \(\alpha \equiv \beta\). We introduce a *counting function* \(c\) such that, given some multiset \(\alpha\), \(c(A;\alpha)\) is the number of times \(A\) occurs in \(\alpha\). Note that \(c(A;\alpha) = 0\) indicates that \(A\) does not occur in \(\alpha\). In similar vein, we let \(\text{crd}\) be a function such that \(\text{crd}(\alpha)\) is the number of formulas including repetitions that occur in \(\alpha\); we say that \(\text{crd}(\alpha)\) is the *cardinality* of \(\alpha\). We are now in a position to define *identity* between multisets. For all multisets \(\alpha, \beta\), \(\alpha = \beta\) iff for all formulas \(A\), \(c(A;\alpha) = c(A;\beta)\).

We will also need to define two notions of multiset containment. A multiset \(\alpha\) is *weakly contained* in a multiset \(\beta\), in symbols, \(\alpha \subseteq \beta\), iff for all formulas \(A\), \(c(A;\alpha) \leq c(A;\beta)\), and secondly, \(\alpha\) is *strongly contained* in \(\beta\), in symbols \(\alpha \triangleleft \beta\), iff (i) \(\alpha \subseteq \beta\) and (ii) \(\alpha \neq \beta\). Finally, we define the concept of *multiset union* which for our purposes we shall represent as \(\cdot\). For all multisets \(\alpha\) and \(\beta\),
\[\alpha \cdot \beta\]
is said to be their multiset union iff (i) \(g(\alpha, \beta) = g(\alpha) \cup g(\beta)\) and (ii) for all \(A\),
\[c(A;\alpha, \beta) = c(A;\alpha) + c(A;\beta)\]. Occasionally we shall have recourse to representing a multiset in exponential notation, by taking the generating set of a multiset \(\alpha\) and placing exponents on its members to indicate the number of times they occur in \(\alpha\). Thus \[[A,B,A,C,A,C]\] may be represented as \([A^3, B, C^2]\). Finally, we define the *power multiset* of a multiset: \(P(\alpha) = \{\beta: \beta \triangleleft \alpha\}\).
Our consecutions then are objects of the form

\[ \vdash \alpha \]

where \( \alpha \) is a multiset of formulas. We will drop multiset brackets where context makes the meaning clear, and we will also suppress the \( \vdash \) symbol - the reader may replace it at the far left of each multiset. As we are working with a right-sided Gentzenization in which only one multiset features in any consecution, we will often refer to a consecution simply as a multiset. We now provide some terminology, freely borrowed from [Curry 63], for presenting our various Gentzen-style formulations of LR.

**Definition 2.2:** In the statement of each connective or structural rule, the upper multiset(s) is (are) called the premiss(es) of the rule and the lower multiset is called the conclusion of the rule. The formula in a conclusion that is newly introduced by a rule is called the **principal constituent** of the rule, and the formula(s) in the premiss(es) that make up the parts of the principal constituent is (are) called the **component(s)** of the rule. Formulas that remain unaltered in a move from premiss(es) to conclusion are called **parametric** constituents of the rule. This terminology is extended as in [Curry 63] to include the contraction rule \( W \) (to be given below), with the formula \( A \) omitted in the move from premiss to conclusion being called a **quasi-parametric** constituent of the rule.

In Table 2-2 we present the rules that define our basic Gentzen formulation of LR, which we call L1. For purposes that will become clear in Section 2.2, we divide the connective rules into two types. Names for each of the rules or axioms occur to the right of each derivation line. Note that although we take the connectives \( \land \) and \( + \) and the constant \( f \) as defined, we also give rules for them in what follows.

The rule \( Kf \) is actually an introduction rule for the constant \( f \) (i.e. \( \sim t \)), but as it is a specialized instance of the classical Gentzen-style structural rule *weakening* (sometimes called *thinning*, but which we will call \( K \) following [Curry 63]) we have classed it amongst our structural rules. Note, however, that the general form of the
Axioms

\[ p, \neg p \quad Axp \quad \text{where } p \text{ is any propositional variable} \]

\[ t \quad T, \alpha \quad AxT \]

Connective Rules - Type 1

\[ \frac{\alpha, A}{\neg \neg A} \quad \frac{\alpha, \neg A}{\neg \neg \alpha} \quad \frac{\alpha, \neg B}{\neg \neg \alpha} \]

\[ \alpha, \neg A, \neg B \quad \alpha, A, \beta, \neg B \quad \alpha, \neg A, \neg B \]

\[ \alpha, \neg A, \beta, \neg B \quad \alpha, A, \beta, \neg B \quad \alpha, \neg A, \beta, \neg B \]

Connective Rules - Type 2

\[ \frac{\alpha, A}{A \lor B} \quad \frac{\alpha, B}{A \lor B} \]

\[ \alpha, A, B \quad \alpha, A \land B \quad \alpha, B \land A \]

\[ \alpha, A, B \quad \alpha, A \lor B \quad \alpha, B \lor A \]

Structural Rules

\[ \frac{\alpha, A, A}{\alpha, A} \quad \frac{\alpha, A}{A} \quad \frac{\alpha}{A} \]

\[ \alpha \quad \text{KI} \quad \alpha, \text{If} \]

Table 2-2: Formulation L1

Defining a system \( L \) be any Gentzen system, formulation of \( L \) is considered to be an immediate consequence of the multi-step propositional rules above, and \( \alpha \) is at the root of the tree. If there is an \( L \)-proof of \( \alpha \) then we shall mean the length of the longest branch in it.

To illustrate we present a Gentzen-style proof, based on formulation L1, of axiom A4. Each \( \vdash \)-proof is annotated with the name of the rule used to derive it.
weakening rule, given in Figure 2-1, is not among the rules of L1. K is paradigmatically an irrelevant rule, and its inclusion would enable a derivation of all classical theorems including the paradoxes.

**Figure 2-1:** The Weakening Rule K

\[ \frac{\alpha}{\alpha, A} K \]

**Definition 2.3:** Let L be any Gentzen-style formulation of LR that we consider in this thesis, including L1. An L-proof of a multiset \( \alpha \) is a binary tree \( \tau \) of multisets such that multisets at the tips of \( \tau \) are axioms, every other multiset follows by the application of exactly one rule of L to an immediately preceding multiset (in the case of one-premiss rules) or pair of multisets (in the case of two-premiss rules), and \( \alpha \) is at the root of the tree. If there is an L-proof of \( \alpha \) then we will say that \( \alpha \) is L-provable. We will let \( \tau, \tau', \tau'' \ldots \) stand for proofs. Moreover, if \( \tau \) is a proof of \( \alpha \), then by the number of steps in \( \tau \) or the length of proof of \( \tau \) we shall mean the length of the longest branch in \( \tau \).

To illustrate we present a Gentzen-style proof, based on formulation L1, of axiom A4. Each step in the proof is annotated with the name of the L1-rule used to derive it.

**Figure 2-2:** The Proof of A4 in L1

\begin{align*}
A, \neg A & \quad \neg B, B \quad P_1 \\
A, \neg A & \quad \neg (A \rightarrow B), \neg A, B \quad P_2 \\
\neg (A \rightarrow A \rightarrow B), \neg A, \neg A, B & \quad W \\
\neg (A \rightarrow A \rightarrow B), \neg A, B & \quad P_3 \\
\neg (A \rightarrow A \rightarrow B), A \rightarrow B & \quad P_4 \\
(A \rightarrow A \rightarrow B) \rightarrow A \rightarrow B &
\end{align*}
**Theorem 2.4:** (Gentzen, Kripke, Belnap and Wallace, Meyer) For all formulas $A$, $A$ is a theorem of $LR$ iff $A$ is $L1$-provable.

**Proof:** The proof of this theorem follows from [Kripke 59], [Belnap and Wallace 65] and [Meyer 66]. Naturally, the soundness half of this proof relies on showing that an appropriate Gentzen-style *Cut rule* for $L1$, given in Figure 2-3, is *admissible* in $L1$ or equivalently, if $L1$ were to be formulated with *Cut* as a primitive rule, that this rule is *eliminable*, without decreasing the stock of $L1$-theorems. Either way, this half of the proof is a straightforward modification of the proof of the *Haupsatz* given in [Gentzen 35]. Q.E.D.

![Figure 2-3: Cut Rule for L1](image)

\[ \alpha, \neg \beta \vdash \alpha, \neg \beta \]

2.1.4. Decidability: formulation $Lk$

In trying to determine if some multiset $\alpha$ is $L1$-provable one strategy is that of [Anderson and Belnap 75] (p.137), say, of taking a proof search tree for $\alpha$. A *proof search tree* for some multiset $\alpha$, or $pst(\alpha)$, is an $n$-ary tree such that every node has a multiset of formulas assigned to it, $\alpha$ is at the root of the tree, and $pst(\alpha)$ has as a subtree a proof of $\alpha$ if $\alpha$ is $L1$-provable. Rather than work from possible premiss(es) to desired conclusion as is implied by Definition 2.3, one constructs a proof search tree for $\alpha$ by applying *in reverse* all of the $L1$-rules that could have $\alpha$ as conclusion, to discover possible premisses which will be the immediate successors of $\alpha$ in $pst(\alpha)$ and which may be used to derive $\alpha$ using the usual unreversed rule. Then in turn, one generates candidate premisses for the candidate premiss(es) for $\alpha$, and so on. There is always a procedure for constructing $pst(\alpha)$, for any finite multiset $\alpha$, because any member of $\alpha$ could only have been introduced, as principal constituent, by the application of one of the $L1$-rules of which their are only finitely many, and likewise for any multiset $\beta$ in $pst(\alpha)$ so constructed. The
question as to whether $L1$ is decidable then amounts, given this strategy, to whether $pst(\alpha)$ is finite.

For reasons that will become clear shortly, we note in passing that $pst(\alpha)$ is indeed an AND/OR tree. For a clear exposition of the basic ideas of AND/OR trees, see [Nilsson 80]. In the case of Gentzen formulations, where we are constructing $pst(\alpha)$ for $\alpha = A$, the initial node in our AND/OR tree is just the multiset containing $A$. This node is expanded to produce immediate successor nodes by applying all permissible Gentzen-style rules in reverse, with each reversed rule application decomposing $\alpha$ into either a single premiss $\beta_1$ or a pair of premisses, $\beta_1$ and $\beta_2$ depending on the rule, from which $\alpha$ may be derived using the usual unreversed Gentzen-style rule. Adapting the terminology of [Nilsson 80], we will say that $\beta_1$ (and $\beta_2$) is (are jointly) AND-wise linked to an immediate predecessor node $\beta$ if $\beta$ can be derived from $\beta_1$ (and $\beta_2$) using a single $L1$-rule; and alternative premiss-sets of AND-linked nodes will be said to be OR-wise linked to $\beta$. By a proof-attempt for or potential proof of some multiset $\alpha$ we shall mean any subtree of AND-linked nodes of $pst(\alpha)$ having $\alpha$ at the root of the subtree. An inspection of Definition 2.3 will show that an $L1$-proof of $\alpha$ is just a particular kind of proof-attempt for $\alpha$. Figure 2-4 illustrates a part of the proof search tree, based on $L1$ rules, for the $LR$ axiom, $A4$. We adopt the conventions of [Nilsson 80] for displaying AND/OR trees:¹ each set of AND-linked nodes have their links arced where necessary to distinguish them from nodes belonging to alternative premiss-sets, and the subtree of $pst(\alpha)$ that constitutes an $L1$-proof of $\alpha$ is emphasized with thick-lined links. Note that the contained proof of $A4$ is just Figure 2-2.

To return to our discussion of $pst(\alpha)$, in general there is no guarantee that $pst(\alpha)$ will be finite, and in the particular case of the system $L1$, the sticking point will be the rule of contraction; i.e. the rule $W$. For a simple example, consider the branch through $pst(\alpha)$ that consists of repeated applications of only the $W$ rule (applied 'in

¹With the exception that from our perspective, all the trees in [Nilsson 80] appear upside-down!
**Figure 2-4:** Initial Segment of a Proof Search Tree

While it is possible to devise a Gentzen formulation of $L_1$ equivalent to $L_1$, it is impossible to devise a Gentzen formulation of $L_2$ equivalent to $L_1$ which contains the structural rule $A'$. We will see shortly, such a move gives us enough controllable advantage to justify tableaux construction in favour of building the effect of $W$ into the connective of the premise. In $L_2$, there is an $A'$ which the structural rule $A'$ does not feature. This formulation of $L_2$ is due to Kripke (90) and Table 2-3 we present a version of $L_2$ in which the rule $W$ does not feature. This formulation of $L_2$ is due to Kripke (90) and Table 2-3 we present a version of $L_2$ in which the rule $W$ does not feature. This formulation of $L_2$ is due to Kripke (90) and...

$L_k$ has the same axioms as $L_1$, and to be more precise structural rule $A'$ for

1. introducing new connectives into $L_k$ are those of $L_1$ modified to include the effect of features from the $L_1$-rules $P_1$, $P_2$, $P_3$, $P_4$, $P_5$, $P_6$, $P_7$, and $P_8$ are modified so as to permit the principal constituent to occur in a premise. The rules $P_9$ and $P_10$ are likewise modified, but if the principal constituent is quasi-parametric in one premise it is required to be quasi-parametric in the other premise. The other connectives rules of $L_k$ permit the built-in contraction on the principal constituent, and limited built-in contraction on parameters. Note that the conditions on $\alpha$, $\beta$ and $\gamma$ in the connective rules of $L_k$ are in terms of the quasi-parametric containment relation $\rightarrow$. They can be stated alternatively, more perspicuously, but not so succinctly in terms of our counting function, $c$ on members of a multiset, as follows: let $\alpha$ be the appropriate principal constituent of any of the above three rules, then

- a quasi-parametric copy of $\alpha$ may occur in $\beta$, or $\gamma$ (if present) or
- neither or both, and

- where $e(D_1) = a$, then $e(D_2) \leq a$ and $e(D_3) \leq a$ and $e(D_4) + e(D_5) \leq a$ (i.e. every member of $\alpha$ must come from at least one of $\beta$ or $\gamma$, or both,
- with this last option amounting to contraction on that member).
reverse’, of course); there will always be such a branch, and it will always be infinitely long.

While contraction is a principle of LR, evidenced by the the contraction axiom A4, it is possible to devise a Gentzen formulation of LR equivalent to L1 in which the structural rule W for contraction is eliminated in favour of building the effect of W into the connective rules. As we will see shortly, such a move gives us enough control over contraction so that the strategy for constructing a decision procedure for L1 suggested in the opening paragraph of this subsection can be effected. In Table 2-3 we present a Gentzen formulation of LR which we call Lk in which the rule W does not feature. This formulation is essentially based on [Kripke 59] and [Curry 50].

Lk has the same axioms as L1, and just the ersatz structural rule Kf for introducing f. The connective rules of Lk are those of L1 modified to include the effect of W. In particular, the L1-rules P, P̅, →, and P+ and Pν are modified so as to permit the principal constituent to occur in a premiss. The rules P & and Pν are likewise modified, but if the principal constituent is quasi-parametric in one premiss it must be quasi-parametric in the other premiss also. The other connective rules of Lk permit built-in contraction on the principal constituent, and limited built-in contraction on parameters. Note that the conditions on α, β and γ in the connective rules of Lk are in terms of our weak multiset containment relation, \(/\). They can be stated alternatively, more perspicuously, but not as succinctly in terms of our counting function, c on members of a multiset, as follows: let pc be the appropriate principal constituent of any of the above three rules, then

- a quasi-parametric copy of pc may occur in β, or γ (if present), or neither or both, and
- where c(D; α) = n, then c(D; β) ≤ n and c(D; γ) ≤ n and c(D; β) + c(D; γ) ≥ n
  (i.e. every member of α must come from at least one of β or γ, or both, with this last option amounting to contraction on that member).
**Table 2-3:** Formulation Lk

**Axioms and Kf-Rule**

$Axp, \ AxT, \ Axt$ and $Kf$ as in Table 2-2.

**Connective Rules - Type 1**

- $\frac{\beta, \ A \quad Pk_\&}{\alpha, \sim \sim A}$
- $\frac{\beta, \sim A \quad Pk_\&}{\alpha, \sim (A \& B)}$
- $\frac{\beta, \sim B \quad Pk_\&}{\alpha, \sim (A \& B)}$
- $\frac{\beta, \sim A, B \quad Pk_\&}{\alpha, A \rightarrow B}$
- $\frac{\beta, A \quad \gamma, \sim B \quad Pk_\&}{\alpha, \sim (A \rightarrow B)}$
- $\frac{\beta, \sim A, \sim B \quad Pk_\&}{\alpha, \sim (A \cup B)}$
- $\frac{\beta, A \quad \sim A \quad \gamma, \sim B \quad Pk_\&}{\alpha, \sim (A \rightarrow B)}$

**Connective Rules - Type 2**

- $\frac{\beta, A \quad Pk_\&}{\alpha, A \vee B}$
- $\frac{\beta, B \quad Pk_\&}{\alpha, A \vee B}$
- $\frac{\beta, A \quad \beta, B \quad Pk_\&}{\alpha, A \& B}$
- $\frac{\alpha, A \cdot B \quad Pk_+}{\alpha, A \& B}$
- $\frac{\beta, A \quad \gamma, B \quad Pk_\&}{\alpha, A \lor B}$

**Conditions:**

In all the connective rules, taking $\delta$ as the conclusion:

(i) $\beta/\delta$,

(ii) $\gamma/\delta$ and

(iii) $\alpha/\beta, \gamma$
Theorem 2.5: (Kripke, Belnap and Wallace, and Meyer) $L_k$ is equivalent to $L_1$.

Proof: That $L_k$-provability implies $L_1$-provability follows by observing that an $L_k$-proof can be transformed into an $L_1$-proof by replacing any tacit built-in contractions used in applying the $L_k$-rules (as described above) with explicit uses of $W$. The converse holds by observing that $L_1$ connective rules are specialized instances of the $L_k$-rules, the ersatz structural rule $K_f$ is common to both formulations, and showing that $W$ is admissible in $L_k$. Proving the admissibility of $W$ in $L_k$ is a straightforward matter; details can be extracted from [Kripke 59], [Belnap and Wallace 61], [Meyer 66] or [Anderson and Belnap 75]. Q.E.D.

Indeed, a result somewhat stronger than the mere admissibility of $W$ in $L_k$ is available, namely:

Lemma 2.6: (Curry) If there is an $L_k$-proof, $\tau$, of $[\alpha, A, A]$, in $n$ steps, then there is an $L_k$-proof, $\tau'$, of $[\alpha, A]$ in no more steps than $n$.

Proof: Can be extracted from [Curry 50]. Q.E.D.

This leads us to the following important definition.

Definition 2.7: We will say that a potential proof $\tau$ of $\alpha$ has the Curry property iff for all multisets $\beta$ and $\gamma$ in $\tau$, if $\gamma$ is a successor of $\beta$ in the sense that $\gamma$ lies in some branch from $\beta$ to a tip of $\tau$, then it is not the case that $\beta | \gamma$ (i.e. put informally, that $\beta$ can be derived from $\gamma$ by a series of contractions or contraction-related moves). We extend this terminology and say that $pst(\alpha)$ has the Curry property iff every potential proof in $pst(\alpha)$ has the Curry property.
Figure 2-5: Examples of the Curry Property

\begin{center}
\begin{tabular}{c|c}
ok & no good \\
\hline
no good & AoB,B,B \\
\hline
AoB,B,B & AoB,A,B \\
\hline
AoB,B & \\
\end{tabular}
\end{center}

The no good nodes, containing [AoB,B,B], are both divided by the multiset at the root, [AoB,B].

**Theorem 2.8:** (Curry) For all multisets \( \alpha \), if \( \alpha \) is Lk-provable then there exists an Lk-proof \( \tau \) of \( \alpha \) that has the Curry property.

**Proof:** Immediate from Lemma 2.6. Q.E.D.

The Curry property can be seen as an extension of an idea which is quite familiar in various strategies for searching partially ordered structures like trees. The idea involves detecting when a newly-generated state description (e.g. a multiset at a node) is identical to a state description that already occurs on the path back to the initial state description (e.g., \( \alpha \) in \( \text{pst}(\alpha) \)). When such a situation occurs, the generation/search procedure fails the current branch, and excludes it from further consideration or expansion. See, for example, [Nilsson 80] (p.25). The Curry property is stronger, in that it will not only debar identical state descriptions from recurring in a given path, but will also exclude those state descriptions that bear the strong multiset containment relation to a multiset that occurs earlier in the tree.

In virtue of Theorem 2.8 we will require that all \( \text{pst}(\alpha) \) have the Curry property. It is important to note that, because of this requirement, proof search trees cannot be represented as AND/OR graphs, but only as the more specialized AND/OR trees. Again, see [Nilsson 80] (p.100) for an explication of this terminology. Many ATP systems can take advantage of the fact that the search space can be regarded...
as a graph - we will not be able to do so. The reason is that AND/OR graphs identify nodes on different paths that have the same state description, thus allowing a node to have more than one immediate predecessor and more than one path to the initial state description, whereas neither of these options is available in the presence of the Curry property. In order to assist the explanation of why this is so, consider the segment of some \( \text{pst}(\alpha) \) based say, on \( \text{Lk} \)-rules, given in Figure 2-6.

**Figure 2-6: AND/OR Graphs and the Curry Property**

\[
\delta = [A,A,C]
\]

\[\uparrow\]

\[
\beta = [A,A,B]
\]

\[\uparrow\]

\[
\epsilon = [A,C]
\]

\[\uparrow\]

\[
\gamma = [A,A,B]
\]

\[\uparrow\]

\[
\alpha
\]

The multisets at nodes \( \beta \) and \( \gamma \), on different branches in \( \text{pst}(\alpha) \) are identical. If we were permitted to represent \( \text{pst}(\alpha) \) as an AND/OR graph then we could identify nodes \( \beta \) and \( \gamma \) and thus identify the subtrees on them; identify \( \delta \) and \( \gamma \) also. The multiset at node \( \delta \) in the subtree on \( \beta \) strongly contains the multiset at node \( \epsilon \) in the branch from \( \beta \) to \( \alpha \), and so because of Curry property considerations cannot be expanded. However, the counter-part of \( \delta \) in the subtree on \( \gamma \) (i.e. \( \gamma \)) does not strongly contain any multiset in its branch from \( \alpha \) and so must be expanded if we are to have a complete proof search tree. If we identify \( \beta \) and \( \gamma \), then we impose inconsistent control conditions on the node and of course this is not permissible.
Naturally, if \( \beta \) and \( \beta' \) were to have exactly the same path, as in the case for example where \( \beta \) and \( \beta' \) are identical premisses of an application of the \( \text{Pko} \) rule, then Curry property considerations could not be violated by identifying them. From a logical point of view identifying \( \beta \) and \( \beta' \) is not significant, and so to keep our discussion of the decision problem as uncomplicated as is possible, we will not take advantage of the limited graphing possibilities that may occasionally arise and identify \( \beta \) and \( \beta' \) where they have identical paths to the root. However, from the perspective of implementing decision procedures, the space savings and control simplifications that can be realized by identifying them can be significant, and for these purposes we will collapse the tree to a (restricted) graph where possible. The details of these restricted graphing possibilities can wait until Chapter 4.

We are now in the position to return to the decision problem for LR. If we can show that \( \text{Lk} \) is decidable, then Theorems 2.4 and 2.5 will take us the rest of the way. We adopt the strategy intimated at the beginning of this subsection. Theorem 2.8 assures us that we can restrict ourselves to searching \( \text{pst}(\alpha) \) for a proof that has the Curry property. So we construct \( \text{pst}(\alpha) \) as before, but this time checking that each multiset introduced into \( \text{pst}(\alpha) \) preserves the Curry property for \( \text{pst}(\alpha) \). Returning to the decision problem, we now have to show that \( \text{pst}(\alpha) \) with the Curry property is finite. We invoke the following lemma due to Konig:

**Lemma 2.9:** (Konig) A tree is finite iff it has the finite branch property (i.e. every branch in the tree contains only finitely many nodes) and the finite fork property (i.e. every node has only finitely many immediate successor nodes).

**Proof:** The proof can be extracted from [Konig 36]. Q.E.D.

That \( \text{pst}(\alpha) \) has the finite fork property follows immediately from an inspection of the rules of \( \text{Lk} \), and the fact that all multisets in \( \text{pst}(\alpha) \) have only finitely many members. The difficult part of the decidability argument comes in establishing that \( \text{pst}(\alpha) \) has the finite branch property, and this involves several more reductions of the decision problem.
Definition 2.10: We now define what it is for one formula to be a subformula (which we abbreviate as sub) of another formula:

- A is a sub of A
- if C is a sub of A then C is a sub of \( \neg A \)
- if C is a sub of A or B then C is a sub of A\&B, AoB, AvB and A+B
- if C is a sub of \( \neg A \) or \( \neg B \) then C is a sub of \( \neg(A\&B) \), \( \neg(AoB) \), \( \neg(AvB) \), \( \neg(A+B) \)
- if C is a sub of \( \neg A \) or B then C is a sub of A\( \rightarrow B \)
- if C is a sub of A or \( \neg B \) then C is a sub of \( \neg(A\rightarrow B) \).

We thus say that a proof \( \tau \) of a multiset \( \alpha \) has the subformula property iff for all formulas A and multisets \( \beta \) in \( \tau \), if A occurs as a subformula of some formula in \( \beta \) then A occurs as a subformula of some formula in \( \alpha \).

Lemma 2.11: (Gentzen) All Lk-proofs have the subformula property.

Proof: Essentially from [Gentzen 35]. Q.E.D.

Given Lemma 2.11, and the obvious fact that the cognation relation, \( \simeq \), between multisets partitions an arbitrary set of multisets into disjoint equivalence classes, it is a straightforward matter to prove the following lemma.

Lemma 2.12: Let \( \kappa \) be some arbitrary branch in \( \text{psl}(\alpha) \) and let \( \mathcal{K} \) be the set of multisets in \( \kappa \). Then \( \simeq \) partitions \( \mathcal{K} \) into a finite number of equivalence classes; i.e. \( \mathcal{K}/\simeq \) is finite.

Proof: The cognation relation will by definition place multisets that have the same generating set into the same equivalence class, so the question becomes one of the upper limit on the number of distinct sets of formulas in \( \text{psl}(\alpha) \). But since \( \text{psl}(\alpha) \) has the subformula property, by Lemma 2.11, and given that \( \alpha \) is a finite multiset of formulas, this upper limit is the power set of the set of subformulas of formulas of \( \alpha \), and hence is finite. Q.E.D.
This reduces the problem of showing that \( \text{pst}(\alpha) \) has the finite branch property to the problem of showing that each member of \( \mathcal{K}/\sim \) is finite. inching our way to the pinnacle, this problem can in turn be reduced further. We say that a sequence of multisets is an \textit{irredundant cognate sequence} iff for all distinct multisets \( \beta \) and \( \gamma \) in this sequence (i) \( \beta \preceq \gamma \) and (ii) if \( \beta \) precedes \( \gamma \) in the sequence then it is not the case that \( \beta \sim \gamma \). So consider some arbitrary equivalence class \( \mathcal{L} \in \mathcal{K}/\sim \). Note that we can order the members of \( \mathcal{L} \) via the natural successor-ordering within the branch \( \kappa \), to get a sequence \( k \). Lemma 2.12 will then guarantee condition (i) and Lemma 2.8 will guarantee that condition (ii) holds of the sequence \( k \) so ordered. So the problem of showing that \( \text{pst}(\alpha) \) has the finite branch property has finally been reduced to showing that all irredundant cognate sequences are finite. This is exactly what Kripke's extraordinary theorem states.

**Theorem 2.13:** (Kripke) For all irredundant cognate sequences \( \mathcal{L} \in \mathcal{K}/\sim \), \( \mathcal{L} \) is finite.

**Proof:** Proved by induction as in, for instance, [Anderson and Belnap 75] (p. 139). Q.E.D.

Although we have omitted details of the proof, we will present a delightfully clear geometric sketch of the proof, from [Dunn 87]. There are several proofs of theorems essentially equivalent to Kripke's theorem to be found in the literature\(^1\), but for those readers who like the intuitive grasp that a picture can sometimes provide, Dunn's account is without equal.

For simplicity, we consider multisets cognate to \([A,B]\]. Each such multiset can be considered a point in the positive quadrant of the following co-ordinate frame, where the \( x \) co-ordinate records \( c(A;\alpha) \) and the \( y \) co-ordinate records \( c(B;\alpha) \), for \( \alpha \) cognate to \([A,B]\).

---

\(^1\)For example, by coding subformulas as prime numbers, multisets as prime multiples, and appealing to a theorem of number theory due to [Dickson 13] concerning sequences of prime multiples, a proof can be extracted from [Meyer 73] to show that \( \text{pst}(\alpha) \) has the finite branch property.
Note that the origin is labelled 1:1 as $[A,B]$ is minimal in the equivalence class of cognate multisets we are considering. Now, let’s try to build an infinite irredundant sequence. Start with an initial finite multiset $\beta$, say $[A,A,A,B,B]$, and from $\beta$ build a multiset with a colossal number of B’s in it, say $\gamma=[A,A,B,B,B,B,B,B,B]$. But in so building $\gamma$ from $\beta$, one has to reduce the number of A’s in $\beta$, say by one. Notice that in placing a point in the graph, we associate with it a shaded area, which confines our sequence construction because no point may be placed within an already shaded area - to do so would imply that the corresponding multiset was redundant (i.e. that its being in $pst(\alpha)$ violated the Curry property). So, to finish the story, what this means is that each new point must march either one unit closer to the x axis or one unit closer to the y axis. Clearly after a finite number of points one or the other of the two axes must be ‘bumped’, and after a short [sic] while the other must be bumped as well. When this happens there is no space left to play without the sequence becoming redundant. The generalization to the case of $n$ [generators] in the

---

1 Pick a couple of million B's if you like, but draw your own graph.
[multiset] to Euclidean n-space is clear (this is with n finite - with n infinite no axis need ever be bumped)." [Dunn 82] (pp.78-79)

**Theorem 2.14:** (Kripke, and Meyer) \( LR \) is decidable.

**Proof:** The general strategy of this subsection, and Kripke's theorem, were initially employed to show that the logics \( R_\rightarrow \) of [Kripke 59] and \( R\neg \) of [Belnap and Wallace 61] and thus the logic \( R_\uparrow \) were decidable. [Meyer 66] adapted the strategy as indicated above to show that \( LR \) (called therein \( R_\neg \)) was decidable. We have mentioned earlier that Meyer has also provided number-theoretic equivalents of Kripke's theorem, and has also provided an algebraic proof of the decidability of \( LR \). Q.E.D.

We turn now to the last part of this section, where we will discuss the computational complexity of the decision procedure just presented.

**2.1.5. Computational Complexity of \( L_k \)**

Theorem-proving for logics is notoriously computationally 'hard'. The theory of computational complexity gives a precise account for what it is for an algorithm to be 'hard', and decision procedures or semi-decision procedures for logics invariably come out badly on that account. We won't survey the theory of computational complexity here; for a good general account see [Garey and Johnson 79], and for the computational complexity of logical theories in particular, see [Ferrante and Rackoff 79]. The following is a very brief summary, extracted from [Ferrante and Rackoff 79], of the relevant notions. Decidable logics give rise to a recursive set consisting just of the theorems of the logic, and the problem of the computational complexity of a logic is just the problem of determining how much time, or memory space, or both is required to decide the membership of this set. This requires having some formal model for computations (usually, a Turing machine model) which can give rise to a metric for computing time or computing space. This metric is then related to some numeric property of members of the set (e.g. length of formula),
and the membership problem is said to be solvable in time (or space) $t(n)$ iff the machine model accepts inputs of length $n$ in at most $t(n)$ time (or space). The notion of computational complexity can apply to a particular decision procedure for a logic (e.g. Kripke's, for LR), which gives rise to an upper bound on the time or space required to solve the membership problem. A lower bound may be obtained by showing that any procedure for solving the membership problem requires at least $t(n)$ time (or space). Thus upper bounds can be said to apply to particular procedures, and lower bounds to the general decision problem. A decision problem (procedure) solvable in time $t(n)$, where $t$ is a polynomial function on $n$, is said to be a *computationally tractable* problem (procedure). The intended inference in this terminology is that an intractable problem or procedure may be too complicated to repay the effort of tackling it. We thus face three questions:

- what is the complexity of the decision problem for LR?
- what is the complexity of the Kripke-Meyer decision procedure?
- and in the event of either or both proving intractable, is ATP for LR viable?

The first question is open, and no one working in the area of relevant logic has, to our knowledge, figured out a way of tackling it. The question of the complexity of the Kripke-Meyer decision procedure is similarly open. In a private communication to McRobbie in 1981 however, Kripke conjectures that the decidability proof is unprovable in elementary recursive arithmetic. If this conjecture is true, then it follows that the procedure cannot be computed within time bounded above by a fixed composition of exponential functions of $n$ - in other words, the procedure is extremely intractable; see [Ferrante and Rackoff 79] (p.5). The last of our three questions looms ominously, but before turning to it we will amass some evidence in support of Kripke's conjecture.

On the face of it, it seems incredible that a decidable propositional logic should have such an horrendous computational complexity as that suggested by Kripke's conjecture. Other well-known propositional logics have comparatively low
computational complexities. For example, [Ladner 77] shows the decision problem for the modal logics $K$, $T$ and $S4$ to have a polynomial space bound, and [Statman 79] gives a similar result for the zero-order intuitionistic logic $J$. We might wonder then what is it about LR or at least the Kripke-Meyer decision procedure for it that could make for such an enormous jump in complexity?

Basically, our problems still stem from contraction, and the necessary absence in relevant logics of weakening principles like $K$. To get some indication of these problems, consider for example the rule for fusion, $Pko$, given in Table 2-3, which permits the effect of contraction on both the principal constituent and the parametric formulas. [McRobbie 79] gives an upper bound on the number of pairs of premisses from which a given multiset, with a fusion formula as principal constituent, can be derived using the rule $Pko$. So we are interested in the number of ways of selecting $\beta$ and $\gamma$ given $\alpha$. We state and justify the upper bound.

**Lemma 2.15:** (McRobbie) The upper bound on the number of premiss-sets for an application of $Pko$ is $4 \times 3^n$ where $n$ is the cardinality of the multiset $\alpha$ given in the statement of the rule.

**Proof:** The conditions on $\beta$ and $\gamma$ are such that each of the $n$ parametric members of the conclusion (i.e. the members of $\alpha$) can be either in $\beta$ or $\gamma$ or both - thus the $3^n$ - and the principal constituent ($AoB$) can be a member of $\beta$ or $\gamma$ or both or neither - thus the multiplication by 4. Q.E.D.

For example, there are 108 possible pairs of premisses from which the multiset $[AoB,C,D,E]$ could have been derived using $Pko$. However, smaller bounds are available if one takes account of repetitions of generators in a multiset; the multiset $[AoB,C,C,C]$, for example, has exactly 40 distinct pairs of $Pko$ premisses, although McRobbie's bound would give an upper bound of 108 again. McRobbie's function can be generalized to give an upper bound on the number of premiss-sets from which any multiset, $\beta$, may be derived:
\[ k \times 4 \times 3^n + 4l + 2m \]

where \( k \) is the number of fusion, negated implication and negated fission formulas in \( \beta \), \( l \) is the number of disjunctions and negated conjunctions in \( \beta \), \( m \) is the number of remaining complex formulas in \( \beta \) and \( n = \text{crd}(\beta)-1 \) (i.e. \( n \) is the number of parameters). Again, depending on the actual members of a given multiset, there may be substantially fewer distinct such premiss-sets. However, the built in contraction permitted on principal constituents spells trouble still. Consider the multiset \([\text{AoB,} \alpha]\). One of its possible pairs of premisses is \([\text{A,AoB,} \alpha]\) and \([\text{B,AoB,} \alpha]\). In this case, the premisses are more complex than the conclusion, and so the number of premiss-sets from which these premisses may have been derived in turn may be greater than the number from which \([\text{AoB,} \alpha]\) could be derived. So the complexity of multisets in a branch of \( \text{pst}(\alpha) \) may grow horrendously until the requirements of the Curry property eventually slow down, and finally terminate, the growth of a branch. It takes little imagination to see that, even with quite simple \( \alpha \), the number of multisets in \( \text{pst}(\alpha) \) can be staggeringly high. Assuming \( \alpha \) to be provable, the number of multisets in the shortest proof of \( \alpha \) can also be extremely high for relatively simple \( \alpha \).

More importantly, the problem of calculating the size of \( \text{pst}(\alpha) \), for given \( \alpha \), is especially aggravated by the fact that the growth of a particular branch of \( \text{pst}(\alpha) \) is only arrested upon either reaching an axiom, a multiset containing no complex formulas (to which no rules apply), or a multiset the presence of which in \( \text{pst}(\alpha) \) constitutes a violation of the Curry property for \( \text{pst}(\alpha) \). And of course, as violations of the Curry property depend on relations between multisets, this last way of arresting the growth of a branch is dependent on the context of \( \text{pst}(\alpha) \). Thus an assessment of the depth at which a violation of the Curry property will occur because of the presence in \( \text{pst}(\alpha) \) of particular multisets is not obviously predictable from an examination of \( \alpha \). It is, indeed, this problem that makes for our difficulties in trying to calculate the computational complexity of the Kripke-Meyer decision procedure, and which provides evidence for Kripke’s conjecture. Without a means for predicting which successor multisets will lead to a violation of the Curry
property, we would seem to have no way of fixing an upper bound on the length of a branch of \( \text{pst}(\alpha) \), and thus no way of knowing how long we might need to labour with contractions. It seems plausible, then, that Kripke's decision procedure is not elementary recursive.

Given the complexities we face, and the perjorative overtones of the term 'intractable', the reader may wonder at our persevering with the development of an ATP system for LR. Chapter 1, of course, mentions grounds for persevering with this task, from the perspectives of both computer science and logic. But principal amongst our reasons for developing KRIPKE must be the motivations of any ATP project, for at worst KRIPKE will not terminate within practical time constraints, and this problem is faced by any significant ATP system.

Although we will continue to live in the shadow of Kripke's conjecture, mainly because we can neither dispense with nor predictably bound the effect of the requirements of the Curry property, we will be able to \textit{appreciably decrease} the number of premisses from which a multiset may have been derived (i.e. significantly improve on the bounds given by McRobbie's function), and the apparent complexity of our task will correspondingly diminished. The next section is devoted to that task.

\subsection*{2.2. Minimal Proof Theory for LR}

Although we now know that \( \text{Lk} \) is equivalent \( \text{LR} \), that we have a decision procedure for \( \text{Lk} \), and that we will be basing KRIPKE on a decidable Gentzen formulation of \( \text{LR} \), this Gentzen formulation will \textit{not} be \( \text{Lk} \). We have something better in mind. Observing that decidability followed by placing constraints and limits on the use of the contraction principle, our quest shall be to place even more stringent constraints on contraction, and thereby decrease the complexity of our theorem-proving task. We shall show that our final Gentzen formulation of \( \text{LR}, \text{L5} \), which places a considerable number of constraints on \( \text{L5} \)-proofs, is equivalent to \( \text{L1} \) and is decidable. The intervening formulations - \( \text{L2}, \text{L3} \) and \( \text{L4} \) - introduce
these constraints gradually, and in a controlled fashion, ensuring that the final, context sensitive, constraints placed on contraction can live together. In the final subsection of this section, we will return to the subject of computational complexity, and re-assess our position.

2.2.1. Formulation L2: normal-forms and other equivalences

We return to our propositional language $PL$, and the Hilbert-style axiomatization provided for $LR$ in Section 2.1.1. We noted there in passing that $T_1$-$T_4$ are theorems of $LR$, and that certain of the connectives could be introduced by definition. We now observe that $LR$ permits replacement of provable equivalents, in the sense that if $\vdash A \leftrightarrow B$, then we may replace an occurrence of $A$ in formula $C$ by $B$, preserving $LR$-provability; see [Anderson and Belnap 75] (pp.93-94), and in particular, [Meyer 66] (p.173) for details. This opens the way for an appropriate normal-form theorem for $LR$ formulas.

Definition 2.16: Let $A$ be a formula of $LR$. Then its negation-normal-form, $nrm(A)$, is an $LR$-formula uniquely given by the following function:

- if $A$ is a literal then $nrm(A) = A$
- if $A$ is a doubly negated formula of the form $\sim \sim B$ then $nrm(A) = nrm(B)$
- if $A$ is an unnegated formula of the form $B \triangle C$, where $\triangle$ is any connective in $\{&, v, o, +\}$, then $nrm(A) = nrm(B) \triangle nrm(C)$
- if $A$ is a negated formula of the form $\sim (B \triangle C)$, where $\triangle$ is any connective in $\{&, v, o, +\}$, then $nrm(A) = nrm(\sim B) \triangle nrm(\sim C)$, where $\triangle$ is the dual of $\triangle$, with $\&$ and $v$ being each others dual, and $o$ and $+$ being each others dual.
- if $A$ is an unnegated implication of the form $B \rightarrow C$ then $nrm(A) = nrm(\sim B) + nrm(C)$
- if $A$ is a negated implication of the form $\sim (B \rightarrow C)$ then $nrm(A) = nrm(B) \circ nrm(\sim C)$. 

To illustrate, the negation-normal-form of
\[ F_1. \quad \neg((p \rightarrow q) + r) \lor (\neg p + (r \rightarrow q)) \]
would be
\[ F_2. \quad ((p \neg q) \lor r) \land (p \neg r \lor q). \]
A formula in negation-normal-form will be said to be in \textit{nrm-form} or a \textit{nrm-formula}.

**Lemma 2.17:** For all formulas \( A \), \( \vdash_{\text{LR}} A \) iff \( \vdash_{\text{LR}} \text{nrm}(A) \)

**Proof:** Substitution of provable equivalents guarantees that substituting parts of \( A \) for provably equivalent parts will preserve provability. Inspection of axiom A6 (and noting that the other direction of A6 is also an LR theorem), definitions Defo and Def+, and theorems T1, T2, T3 and T4, and the function \( \text{nrm} \) given above, will show that the function \( \text{nrm} \) applied to a formula \( A \) will always have a formula provably equivalent to \( A \) as value. Q.E.D.

**Lemma 2.18:** For all formulas \( A \), (i) \( A \) is \( \text{L1} \)-provable iff \( \text{nrm}(A) \) is \( \text{L1} \)-provable, and (ii) if \( \tau \) is a proof of \( \text{nrm}(A) \) then the only rules used in the construction of \( \tau \) are \( W, P_0, P_+ \), \( P_\lor \), \( P_\land \) and \( Kf \) (i.e. the Type 1 connective rules of \( \text{L1} \) do not feature).

**Proof:** Part (i) follows immediately from Theorem 2.4 and Lemma 2.17. The proof of part (ii) is straightforward. Given Definition 2.10, inspection of how the function \( \text{nrm} \) is defined shows that every complex subformula of \( \text{nrm}(A) \) is of the form \( B \lor C \), \( B \land C \), \( B + C \) and \( B \cap C \). And given that it follows easily from Theorem 2.5 and Lemma 2.11 that \( \text{L1} \) has the subformula property, these formulas could only have been introduced as subformulas of formulas in multisets in \( \tau \) by application of \( P_0, P_+, P_\lor \) and \( P_\land \). Clearly \( W \) and \( Kf \) can also be used in such proofs. Q.E.D.
Definition 2.19: (Formulation L2). We now define formulation L2 as follows: L2 has all the rules and axioms of L1, except the Type 1 connective rules, and only these rules.

Theorem 2.20: L2 is equivalent to L1.

Proof: Immediate from Lemma 2.18, Definition 2.19 and by using the obviously defined translation function. Q.E.D.

This theorem will simplify our move from L1 to L5, if only by decreasing the number of cases we will need to inspect in proving the various formulations equivalent. Moreover, as we have noted previously, all the binary connectives of an nrm-formula are provably associative and commutative, and the conjunction and disjunction connectives are provably idempotent and absorptive (being truly the analogues of the lattice operations for meet and join). These facts open the way for further simplifications of formulas to be put to KRIPKE. We will pursue this matter further in Section 2.3.1 Note especially that none of the normal-forms appealed to are of any logical necessity: we could conduct all the business of this thesis in terms of the full propositional language PL, and KRIPKE accepts any formula of PL as legal input, with internal normal-forming being conducted merely for reasons of runtime efficiency.

We now turn to the proper business of this section, namely, the further control of the contraction principle, W:

---

1Note, however, that in the move from L1 to L2 we have not simplified the 'real' complexity of our task, as discussed in Section 2.1.5. Our translation of formulas into nrm-form involved the substitution of subformulas for equally complex formulas.
2.2.2. Formulation L3: no contraction on + or &.

We formulate L3 with exactly the connective rules and axioms of L2, and the rule Kf, but we replace the structural rule for contraction with *specialized* instances of it. Rather than the full W rule, L3 has:

**Table 2-4: Contraction Rules in L3**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha, \langle 1 \rangle ) ( w_1 )</td>
<td>( \alpha, \langle A \lor B, A \lor B \rangle ) ( w_0 )</td>
</tr>
</tbody>
</table>

where \( \langle \rangle \) is any literal.

Note in particular that L3 does *not* postulate the remaining instances of the W rule:

\[ \alpha, A \lor B, A \lor B \text{ } w_+ \]

\[ \alpha, A \lor \neg B, A \lor B \text{ } w_x \]

\[ \alpha, A \lor B \text{ } w_x \]

\[ \alpha, A \lor B \text{ } w_x \]

We will prove L3 equivalent to L2 by showing that the full structural rule W is admissible in L3. For this we will require two theorems regarding L3-proofs.

**Theorem 2.21:** If \( \alpha = \langle A \lor B, \beta \rangle \) is L3-provable, then \( \alpha' = \langle A, B, \beta \rangle \) is L3-provable, in no more steps.

**Proof:** (We refer the reader back to Definition 2.3 for the intent of the condition 'no more steps'). Our proof will be by induction on \( n \), the length of proof of \( \alpha \). If \( n = 1 \) we have \( \alpha \) as an axiom, and so, in the case of \( A x T \) and \( A x p \), not of the form \( \langle A \lor B, \beta \rangle \). The theorem holds for \( n = 1 \), in these cases, by falsity of antecedent. In the case of \( A x T \), \( \beta \) contains \( T \), and so \( \alpha' \) will contain \( T \), and thus be an axiom. So assume as the inductive hypothesis that the theorem holds for \( < n \); we show it holds for \( n \). We have two cases to consider: \( A \lor B \) is principal constituent in \( \alpha \), or \( A \lor B \) is parametric in \( \alpha \).
Case 1: A+B is principal constituent in \( \alpha \). Then it was derived by \( P^+ \), with \( \alpha' \) as premiss, with \( \alpha' \) obviously having a proof in \( n-1 \) steps. (Note that A+B could not have been introduced any other way, because in particular, L3 is not formulated with \( W^+ \)).

Case 2: A+B is parametric in \( \alpha \). Then \( \alpha \) came by some rule of L3, other than \( P^+ \) with A+B as principal constituent. We have eight subcases to consider, corresponding to the rules of L3.

Case 2.1: \( \alpha = [A+B, CoD, \beta, \gamma] \), and came via an application of \( Po \), with CoD as principal constituent. Without loss of generality, we can consider \( \alpha \) to have been derived via

\[
\alpha_1 = C, A+B, \beta \quad \alpha_0 = D, \gamma \quad P_0
\]

\( \alpha \)

But on the inductive hypothesis, \( \alpha_1 = [C, A, B, \beta] \) is provable in no more steps than \( \alpha_1 \). Thus we have

\[
\alpha_1' = C, A, B, \beta \quad \alpha_2 = D, \gamma \quad P_0
\]

\( \alpha' = A, B, CoD, \beta, \gamma \)

and we have a proof of \( \alpha' \) in no more steps than \( \alpha \).

Cases 2.2 - 2.5:

corresponding to \( Pv, P&, P^+ \) and \( Kf \), have trivial proofs in the fashion of Case 2.1, and we leave these cases to the reader.

Cases 2.6 - 2.8:

corresponding to \( Wl, Wo \) and \( Wv \). Without loss of generality, we assume that \( \alpha = [A+B, C, \beta] \), where \( C \) is either a literal, or of the form (DoE) or (DvE), and came from \( \alpha_1 = [A+B, C, C, \beta] \) by the appropriate instance of \( Wl, Wo \) or \( Wv \). But the inductive hypothesis gives us that \( \alpha_1' = [A, B, C, C, \beta] \) is provable in no more steps than \( \alpha_1 \). But \( \alpha' \) will follow from \( \alpha_1' \) by the same instance of contraction, and will thus be provable in no more steps than \( \alpha \) as required. Q.E.D.
Theorem 2.22: If \( \alpha = [A \& B, \beta] \) is L3-provable, then \( \alpha' = [A, \beta] \) and \( \alpha'' = [B, \beta] \) are each L3-provable in no more steps.

Proof: The proof proceeds in similar manner to that for Theorem 2.21. We give the cases corresponding to 2.1 and 2.6 - 2.8 of that proof.

Case 2.1: \( \alpha = [A \& B, CoD, \beta, \gamma] \), and was derived by Po, with CoD as the principal constituent. Without loss of generality, we can consider \( \alpha \) to have come by

\[
\alpha_1 = C, A \& B, \beta \quad \alpha_2 = D, \gamma \quad P_0
\]

But on the inductive hypothesis, \( \alpha_1' = [C, A, \beta] \) and \( \alpha_1'' = [C, B, \beta] \) are each L3-provable in no more steps than \( \alpha_1 \). Thus we have both

\[
\alpha_1' = C, A, \beta \quad \alpha_2 = D, \gamma \quad P_0
\]

\( \alpha' = A, CoD, \beta, \gamma \)

and

\[
\alpha_1'' = C, B, \beta \quad \alpha_2 = D, \gamma \quad P_0
\]

\( \alpha'' = B, CoD, \beta, \gamma \)

which will each be provable in no more steps than \( \alpha \).

Case 2.6 - 2.8:

\( \alpha = [A \& B, C, \beta] \), where C is either a literal or of the form (DoE) or (DvE), and came from \( \alpha_1 = [A \& B, C, \beta] \) by the appropriate instance of contraction. But on the inductive hypothesis, \( \alpha_1' = [A, C, \beta] \) and \( \alpha_1'' = [B, C, \beta] \) are each L3-provable in no more steps than \( \alpha_1 \). Thus \( \alpha' = [A, C, \beta] \) and \( \alpha'' = [B, C, \beta] \) derivable from \( \alpha_1' \) and \( \alpha_1'' \) respectively, and thus L3-provable in no more steps than \( \alpha \), as required. Q.E.D.

Definition 2.23: We define the degree, \( d(A) \), of a formula, A, recursively as follows: literals have degree 0, and if A is of the form (BoC), (BvC), (B&C), or (B+C), then \( d(A) = d(B) + d(C) + 1 \).

Lemma 2.24: The rule W is admissible in L3.
Proof: We have to show that if \( \alpha = [A,A,\beta] \) is L3-provable then \( \alpha' = [A,\beta] \) is L3-provable. We proceed by induction on the degree of complexity of A. If \( d(A) = 0 \) then A is a literal, and we can use \( Wl \) to derive \( \alpha' \). So assume the lemma holds for \( d(A) < n \); we show it holds for \( n+1 \). Note that if the main connective of A is o or v we can always use \( P_0 \) or \( P_v \) to derive \( \alpha' \). So we only have to show that the lemma holds when either A is a fission or conjunction, of degree \( n+1 \).

Case 1: \( \alpha = [B+C,B+C,B,\beta] \), with \( B+C \) of degree \( n+1 \). But Theorem 2.21 guarantees that \( \alpha_1 = [B,\beta,B,\beta,B] \) will be L3-provable. And \( d(B) \) and \( d(C) \leq n \), so on the inductive hypothesis we have \( \alpha_1' = [B,\beta,B] \), and using \( P_+ \) we can derive \( \alpha' = [B+C,\beta] \) from \( \alpha_1 \).

Case 2: \( \alpha = [B&C,B&C,B,\beta] \), with \( B&C \) of degree \( n+1 \). But Theorem 2.22 assures us that \( \alpha_1 = [B,\beta,B&C] \) and \( \alpha_2 = [\beta,B&C,\beta] \), and thereby that \( \alpha_3 = [B,\beta,B] \) and \( \alpha_4 = [B,C,\beta] \) and \( \alpha_5 = [C,\beta,C] \) will all be L3-provable. And \( d(B) \) and \( d(C) \leq n \), so on inductive hypothesis \( \alpha_3' = [B,\beta,B] \) and \( \alpha_5' = [C,\beta,C] \) will, in particular, be L3-provable. But then we can use \( P_\& \) to derive \( \alpha' = [B&C,\beta] \) from \( \alpha_3' \) and \( \alpha_5' \). Q.E.D.

Theorem 2.25: L3 is equivalent to L2.

Proof: Inspection of the rules will show that L3 is contained in L2. If \( \alpha \) is L2-provable, then as L2 and L3 only differ over instances of the rule \( W \), Lemma 2.24 assures us that \( \alpha \) will be L3-provable. Q.E.D.

We have thus shown that, with respect to fission and conjunction formulas, no contraction need take place. As we shall see shortly, further optimization of the rules \( P_+ \) and \( P_\& \) is available.
2.2.3. Formulation L4: invertible rules and normalized proofs.

In the literature, there is the notion of a rule being invertible or permutable, in the sense that if $\alpha = [\beta, A, B]$ is a multiset with A and B being complex formulas, and $\alpha$ was derived by a rule $P_y$ with A as principal constituent, then $\alpha$ could have been derived by a rule $P_z$ with B as principal constituent. In this case, $P_z$ is said to be invertible with respect to $P_y$. The particular interest in this notion is when $P_y$ and $P_z$ are different rules, and $\alpha$, A and B are thoroughly general, save for the fact that A is introduced by $P_y$ and B by $P_z$; see especially [Curry 50] and [Kleene 52].

Invertibility of rules often gives rise to several important corollaries. For example, if a particular rule, say $P_z$, is invertible with respect to any rule $P_y$ of the particular formulation then it is often possible to show that if $\alpha$ is provable there is a proof of $\alpha$ in which instances of $P_z$ are applied last (traversing the proof from tips to root, in Gentzen fashion). In the context of constructing proof search trees, such corollaries pan out nicely in terms of the tentative/irrevocable distinction made between control strategies in [Nilsson 80] (p.21). Irrevocable strategies are those in which the application of a rule, to expand some node, does not prejudice solution of the problem - i.e. if there is a solution, then there is a solution that begins with the given rule application. With tentative strategies, particular rule applications may prejudice the solution - for example $\alpha = [AvB, CoD, \beta]$ may be L3-provable only if CoD is taken as principal, and so any attempt to prove $\alpha$ by taking AvB as principal will prejudice our ability to in fact prove $\alpha$. [Nilsson 80] (p.36) notes that certain rule-based systems, which he generically terms "commutative production systems", can always have an irrevocable control regime imposed upon them. Later on, on page 163, Nilsson points out that systems for producing resolution-style refutations, within classical logic, are commutative production systems and as such can have irrevocable control strategies imposed on them. This result is equivalent to various well-known results in logic (e.g. [Curry 50]) that Gentzen formulations for TV exist in which all rules are invertible. Curry's proof relies, in part, on building in the effect of the rule of weakening, $K$, (which is admissible in classical logics) into the other rules of TV. As we have previously stated, however, $K$ is
especially destructive of relevant sensibilities, and is definitely not admissible in any relevant logic. We should not expect, therefore, to be able to show that all the rules of any system of relevant logic are invertible, and as a consequence, we will always be saddled with having to employ the tentative control strategies explicit in the proof search tree approach. This is, indeed, one of the reasons we gave in Section 2.1.5 to explain the apparent complexities of the Kripke-Meyer decision procedure (although we stated it somewhat differently at the time).

We will, however, show that the rules \( P^+ \) and \( P^\& \) are invertible rules with respect to the rules of \( L_3 \). Thus later, in Chapter 4, when we come to implementing our decision procedure for \( LR \), we will be able to make somewhat less tentative strategic choices in constructing \( pst(\alpha) \). In particular, we now show that if a multiset \( \alpha \) explicitly contains a formula, \( A \), which would be principal under \( P^+ \) or \( P^\& \), then we need only construct that part of \( pst(\alpha) \) that has \( A \) as principal.

**Corollary 2.26:** If \( \tau \) is an \( L_3 \)-proof of \( \alpha = [A+B, C, \beta] \), with \( C \) as principal constituent, then there is an \( L_3 \)-proof, \( \tau' \), of \( \alpha \) with \( A+B \) as principal constituent.

**Proof:** Theorem 2.21 assures us that \( \alpha^{'} = [A, B, C, \beta] \) will be \( L_3 \)-provable, and using \( \alpha^{'} \) as premiss, we can derive \( \alpha \) using \( P^+ \). Q.E.D.

**Corollary 2.27:** If \( \tau \) is an \( L_3 \)-proof of \( \alpha = [A\&B, C, \beta] \), with \( C \) as principal constituent, then there is an \( L_3 \)-proof, \( \tau' \), of \( \alpha \) in which \( A\&B \) is principal.

**Proof:** As for Corollary 2.26, using Theorem 2.22. Q.E.D.

These corollaries correspond to part of our idea of an invertible rule. Unfortunately, the rules of \( L_3 \) do not state, within themselves so to speak, that \( P^+ \) and \( P^\& \) are invertible, and the invertibility of these rules remains a meta-fact about \( L_3 \). To be exact, it is thus still permissible in \( L_3 \) to have a proof of say \( [A+B, C, \beta] \) in which \( A+B \) is not the principal constituent. As part of our task - if we are to end up in \( L_5 \) with a decidable formulation - will be to show that an
appropriate version of Curry's lemma, Lemma 2.6, is provable, we will face problems in showing that if $\alpha$ is provable then there is a proof, $\tau$, of $\alpha$ which has both an appropriate version of the Curry property and is an inverted proof of $\alpha$. To circumvent this problem, we will build the fact that $P+$ and $P\&$ are invertible into the statement of the rules of L3, to produce formulation L4. To pave the way for L4, we will need some more machinery.

**Definition 2.28:** We define the outer degree, $od(A)$, of a formula $A$ recursively as follows: if $A$ is a literal, or a complex formula with $o$ or $v$ as the main connective, then $od(A) = 0$; otherwise, $A$ is of the form $(B+C)$ or $(B&C)$, and $od(A) = od(B) + od(C) + 1$. We extend this definition to cover multisets of formulas: the outer degree, $od(\alpha)$, of a multiset $\alpha$, is the sum of the outer degrees of its members.

**Definition 2.29:** Let $A$ be any formula, the main connective of which is either a fission or conjunction connective. Then $A$ is said to be an invertible formula. Let $\tau$ be an L3-proof of $\alpha$, where $\alpha$ was derived by some rule $Px$. Then that application of the rule $Px$ is said to be a normal inference for $\alpha$ iff: if $\alpha$ contains invertible formulas (i.e. fission or conjunction formulas) then $\alpha$ was derived using $P+$ or $P\&$, as appropriate. Note that if $\alpha$ contains no invertible formulas, then every application of some rule to produce $\alpha$ will be a normal inference, by falsity of the antecedent of the condition. We will call $\tau$ a normalized proof of some multiset $\alpha$ iff every multiset $\beta$ in $\tau$ came by way of a normal inference in $\tau$, or is an axiom.

**Theorem 2.30:** If $\alpha$ is L3-provable, then there is a normalized L3-proof of $\alpha$.

**Proof:** The proof proceeds by double induction, with the outer induction being on $n$, the length of the proof of $\alpha$, and the inner induction being on $k$, the outer degree $od(\alpha)$ of $\alpha$.

**Outer Induction:**

if $n=1$ then $\alpha$ is an axiom, and its proof is trivially normalized. So assume, as the outer inductive hypothesis, that the theorem holds for $\alpha$
with length of proof less than \( n \); we show it holds for \( n \). We have two general cases to consider: either the principal constituent of \( \alpha \) is an invertible formula, or it is not.

**Case 1:** the principal constituent of \( \alpha \) is an invertible formula. Then, by Definition 2.2g, \( \alpha \) always has a normal inference. But on the outer inductive hypothesis, the premisses of \( \alpha \), each with proofs of length \(< n\), will have normalized proofs, and so \( \alpha \) will.

**Case 2:** the principal constituent of \( \alpha \) is of the form \((A \lor B)\), \((A \land B)\), or a literal, and was derived using \( P_0, P_v, W_l, W_0, W_v \) or \( K_f \) as appropriate.

**Inner Induction:**

If \( od(\alpha) = 0 \), then by Definitions 2.28 and 2.29, \( \alpha \) has no invertible formulas as members, and by Definition 2.29, will thus always have a normal inference. But on the outer inductive hypothesis, the premisses of \( \alpha \) will have normalized proofs since their length of proof is \(< n\), and hence so will \( \alpha \). So assume, given the outer inductive hypothesis, that the theorem holds for \( od(\alpha) < k \). We shall now show that it holds for \( od(\alpha) = k \). For the argument we thus assume, without loss of generality, that \( \alpha \) contains at least one invertible formula, and \( od(\alpha) = k \). Hence irrespective of how \( \alpha \) was derived, it contains an invertible formula as parametric, and is thus of the form \( [C+D, \beta] \) or \( [C&D, \beta] \).

**Case 2.1:** \( \alpha \) is of the form \( [C+D, \beta] \), with \( C+D \) parametric. But Theorem 2.21 assures us that \( \alpha' = [C,D,\beta] \) will be \( L_3 \)-provable in no more steps than \( \alpha \) (i.e. \( n \) steps) and inspection of \( \alpha' \) shows that \( od(\alpha') = k-1 \). Thus, on the inner inductive hypothesis, \( \alpha' \) will have a normalized proof. We may then use \( P_+ \) to derive \( \alpha \) from \( \alpha' \), and as this is a normal inference, \( \alpha \) will thus have a normalized proof.
Case 2.2: $\alpha$ is of the form $[[C&D,\beta]]$, with C&D parametric. Using Theorem 2.22 and the inner inductive hypothesis, and proceeding in similar fashion to Case 2.1, $\alpha$ will thus have a normalized proof. Q.E.D.

We are now in a position to define our next Gentzen formulation of LR, L4. L4 has exactly the axioms and rules of L3, with the exception that the connective rules $P_0$ and $P_\lor$ are replaced by $P_\forall$ and $P_\lor'$ as follows:

Table 2-5: The $P_\forall$ and $P_\lor'$ Rules of L4

<table>
<thead>
<tr>
<th>A,$\alpha$</th>
<th>B,$\beta$</th>
<th>$P_\forall$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_\land B,\alpha,\beta$</td>
<td>A,$\alpha$</td>
<td>$P_\lor'$</td>
</tr>
<tr>
<td>$A_\lor B,\alpha$</td>
<td>$A_\lor B,\alpha$</td>
<td></td>
</tr>
</tbody>
</table>

where for all rules neither $\alpha$ nor $\beta$ contain invertible formulas.

Theorem 2.31: L4 is equivalent to L3.

Proof: The rules of L4 are specialized instances of the L3-rules, and so if a multiset $\alpha$ is L4-provable, it will clearly be L3-provable. For the other direction, it suffices to note that Theorem 2.30 assures us that every L3-provable multiset has a normalized proof, and inspection of Definition 2.29 and the rules $P_\forall'$ and $P_\lor'$ will show that every normalized L3-proof of $\alpha$ will be an L4-proof of $\alpha$. Q.E.D.

2.2.4. Formulation L5: contraction confined; decidability.

In this subsection, we introduce our final formulation, L5, and prove that the rules $W_l$, $W_0$ and $W_\lor$ are eliminable by building their effect into the connective rules. We thereby derive a restricted version of Curry's lemma (Lemma 2.6) and show that L5 is equivalent to L1. Using this limited version of Curry's lemma, we modify the Kripke-Meyer argument, to show L5 decidable.
Table 2-6: Formulation L5

Axioms

\[ p, \sim p \quad \text{Ax}_p \]

where \( p \) is any propositional variable

Structural Rule

\[ \alpha, KI \]

\[ \alpha, f \]

Connective Rules

\[ \alpha, \neg \neg \beta \quad \text{P}_+ \]

\[ \alpha, \neg \beta \quad \text{P}_\& \]

\[ \beta, A \quad \text{P}_" \]

\[ \alpha, A \vee B \]

\[ \beta, B \quad \text{P}_" \]

\[ \alpha, A \neg \vee B \]

Where for each of \( \text{P}_" \)

(A) every member of \( \alpha \) is non-invertible

(B) if \( c(A \vee B; \alpha) = 0 \) then either

(a) \( \beta = \alpha \), or

(b) \( \beta = [\alpha, A \neg \vee B] \)

(C) if \( c(A \vee B; \alpha) > 0 \) then \( \beta = \alpha \)
Connective rules: continued

\[ \beta, A \xrightarrow{\gamma} B \quad P_{\beta\gamma} \]
\[ \alpha, AoB \]

Where the conditions on this rule are

(A) every member of \( \alpha \) is non-invertible

(B) for any formula \( C \) in \( \alpha \)

(a) if \( c(C;\alpha) = 1 \) then either

(i) \( c(C;\beta) = 1 \) and \( c(C;\gamma) = 0 \), or
(ii) \( c(C;\beta) = 0 \) and \( c(C;\gamma) = 1 \), or
(iii) \( c(C;\beta) = 1 \) and \( c(C;\gamma) = 1 \)

(b) if \( c(C;\alpha) > 1 \) then \( c(C;\alpha) = c(C;\beta) + i(c)(C;\gamma) \)

(C)

(a) if \( c(AoB;\alpha) = 0 \) then either

(i) \( c(AoB;\beta) = c(AoB;\gamma) = 0 \), or
(ii) \( c(AoB;\beta) = 1 \) and \( c(AoB;\gamma) = 0 \), or
(iii) \( c(AoB;\beta) = 0 \) and \( c(AoB;\gamma) = 1 \), or
(iv) \( c(AoB;\beta) = c(AoB;\gamma) = 1 \)

(b) if \( c(AoB;\alpha) = 1 \), then as for (B)(a)

(c) if \( c(AoB;\alpha) > 1 \), then as for (B)(b).
Note that there are no structural rules apart from the ersatz rule, $Kf$, for introducing $f$, and that we have only built the effect of contraction into the rules $Po''$ and $Pv''$. In particular, the contraction-related conditions are (B)(b) for $Pv''$, and (B)(a)(iii), (C)(a) and (C)(b) for $Po''$. Conditions (C)(b) and (C)(c) of the $Po''$ are in fact redundant, because the formula $C$ in condition (B) can be $AoB$.

Moreover, note that the contraction-related conditions are stronger than those of the $Lk$-rules $Pko$ and $Pkv$, and limit built-in contraction severely if a formula occurs more than once in the conclusion. The (A)-conditions on $Pv''$ and $Po''$ are just the conditions on the $L4$-rules $Pv'$ and $Po'$ that guarantee normalized proofs.

Our immediate business is to show that the rules $Wl$, $Wo$ and $Wv$ are all admissible in $L5$.

**Theorem 2.32:** If $\tau$ is an $L5$-proof of $\alpha = [A,A,\beta]$, where $A$ is of the form (BoC), (BvC) or a literal, then there is an $L5$-proof $\tau'$ of $\alpha' = [A,\beta]$, in no more steps than $\tau$.

**Proof:** Our proof will be by induction on $n$, the length of proof of $\alpha$. If $n=1$, then $\alpha$ is an axiom, and the theorem will hold by falsity of antecedent, for $Ax, p$ and $Ax, T$. So assume, as the inductive hypothesis, that the theorem holds for proofs of length less than $n$; we show it holds for $n$. We have five main cases to consider, depending on which $L5$-rule was used to derive $\alpha$.

Case 1: $\alpha$ was derived using $P+$, and as $A$ cannot be a fission formula, $\alpha$ is of the form $[A,A,D+E,\gamma]$, with $D+E$ principal. It could only have come from $\alpha_1 = [A,A,D,E,\gamma]$, which has a proof of length less than $n$. By the inductive hypothesis, $\alpha_1' = [A,D,E,\gamma]$ will have a proof of length less than $n$, and so by using $P+$, $\alpha' = [A,D+C,\gamma]$ will be $L5$-provable in no more than $n$ steps.

Case 2: $\alpha$ was derived using $P&$. The case proceeds in similar fashion to Case 1.

Case 3: $\alpha$ was derived using $Pv''$. A may be of the form $DvE$ and so may be
principal. We have two general subcases, depending on whether or not A is principal constituent.

Case 3.1: A is the principal constituent, and thus $\alpha$ is of the form $[D_vE, D_vE, \beta]$, and hence condition (C) of $P_v''$ applies. So $\alpha$ was derived from $\alpha_1 = [D_vE, D, \beta]$ (or $\alpha_2 = [D_vE, E, \beta]$), but this case is symmetric). If $D_vE$ is not in $\beta$, then we may use $P_v''$ under condition (B)(b) to derive $\alpha' = [D_vE, \beta]$ from $\alpha_1$ in no more than $n$ steps. If $D_vE$ is in $\beta$ then $\alpha_1$ is of the form $[D_vE, D_vE, D, \gamma]$ where $\beta = [D_vE, \gamma]$, and as $\alpha_1$ has a proof of length less than $n$, the inductive hypothesis will give us that $[D_vE, D, \gamma]$ has a proof of length less than $n$. Whence, using $P_v''$ under condition (B)(a), we can derive $[D_vE, D_vE, \gamma] = [D_vE, \beta] = \alpha'$ from $\alpha_1$ in no more steps than $n$.

Case 3.2: A is not the principal constituent, and so $\alpha$ is of the form $[A, A, D_vE, \gamma]$, with $D_vE$ principal. Whatever form of the $P_v''$ rule was used, both A’s will be parametric in the premiss and hence we may use the inductive hypothesis to show that this premiss less one copy of A is provable in no more steps than the premiss, and then use the same form of the $P_v''$ rule to derive $\alpha''$ in no more than $n$ steps.

Case 4: $\alpha$ was derived using $P_v'$'. A may be of the form $D_oE$ and so may be principal. We have two general subcases, depending on whether or not A is the principal constituent.

Case 4.1: A is the principal constituent, and thus $\alpha$ is of the form $[D_oE, D_oE, \beta]$. Let $\delta = [D_oE, \beta]$, the multiset of parametric members of $\alpha$. Note that as $\delta$ has at least one copy of the principal constituent, condition (C)(a) of the $P_v'$ rule cannot apply here to constrain the premisses. If $c(D_oE; \delta) = 1$, then condition (B)(a) (via (C)(b)) applies, and if $>1$ then condition (B)(b) (via (C)(c)) applies.
Case 4.1.1:
Assume \( c(\text{DoE};\delta) = 1 \). But the conditions on the occurrence of \( A \) in the premisses for \( \alpha \), under \((B)(b)\), are a subset of the conditions on premisses under \((C)(a)\) - i.e. they are exactly \((C)(a)(\text{ii})-(\text{iv})\) - and hence the \( P^\sigma \) rule can be applied with these premisses to constrain the occurrence of \( A \) in the conclusion, so that \( c(\text{DoE};\delta) \) can equal 0 while the count of other members remains unaltered. But then among the multisets that can be derived in one step from these premisses for \( \alpha \) is exactly \( c_1 \); whence \( c_1 \) will be provable in no more steps than \( \alpha \).

Case 4.1.2:
Assume \( c(\text{DoE};\delta) > 1 \); hence the occurrence of \( \text{DoE} \) in the premisses is constrained by condition \((B)(b)\). The interesting subcase is where \( \text{DoE} \) occurs at least twice in at least one of the premisses, whence the inductive hypothesis will deliver the theorem in the manner of Case 3.2, using \( P^\sigma \) under condition \((B)(b)\). The other subcase is where \( \text{DoE} \) occurs exactly once in each premiss. But in this case we can take the occurrence of \( \text{DoE} \) in the premisses as being constrained under condition \((B)(a)(\text{iii})\), and derive \( \alpha' \) from these premisses in no more steps than our derivation of \( \alpha \).

Case 4.2: A is not principal, and thus occurs at least twice parametrically in \( \alpha \), whence condition \((B)(b)\) applies. If \( A \) occurs exactly once parametrically in each of the the premisses for \( \alpha \), then we can derive \( \alpha' \) from these premisses under condition \((B)(a)(\text{iii})\). Otherwise \( A \) occurs at least twice in one of the premisses, and the argument proceeds in the manner of Case 3.2.

Case 5: \( \alpha \) was derived using \( Kf \), and so is of the form \([A,A,f,\gamma]\), where \( f \) is principal. The case proceeds as for Case 1 above. Q.E.D.

**Theorem 2.33:** \( L_5 \) is equivalent to \( L_1 \).
Proof: Theorems 2.20, 2.25 and 2.31 give us the fact that $L_1$ is equivalent to $L_4$, so we show $L_4$ equivalent to $L_5$. If $\tau$ is an $L_5$-proof of $\alpha$, then we will be able to derive $\alpha$ in $L_4$ by using explicit contraction moves (i.e. $Wl$, $Wv$ or $Wo$) to replace the implicit contractions in $\tau$ built into the $L_5$ rules. For the other direction, if $\alpha$ is $L_4$-provable, then as the $L_4$ connective rules are contained in their $L_5$ correlatives, and as $Wl$, $Wo$ and $Wv$ are admissible in $L_5$ by Theorem 2.32, $\alpha$ must be $L_5$-provable. Q.E.D.

Definition 2.34: Let $\tau$ be an $L_5$-proof of $\alpha$. Then $\tau$ is said to have the restricted Curry property iff, for all multisets $\beta$ and $\gamma$ in $\tau$, if neither $\beta$ nor $\gamma$ contain invertible formulas, then if $\gamma$ is a successor of $\beta$ then it is not the case that $\beta | \gamma$. This is just our definition of the Curry property in Section 2.1.4, restricted in accord with the limitations of Theorem 2.32.

Lemma 2.35: For all multisets $\alpha$, if $\alpha$ is $L_5$-provable then there is an $L_5$-proof $\tau$ of $\alpha$ that has the restricted Curry property.

Proof: Immediate from Theorem 2.32.

Theorem 2.36: $L_5$ is decidable.

Proof: The proof follows that of Section 2.1.4 for $L_k$. In particular, we use Lemma 2.35 and adapted versions of the subformula property and the Cognition Lemma, both of which are trivially provable for $L_5$. The difficulty arises in showing that Lemma 2.35 suffices. Lemma 2.35 only assures us that some of the cognate sequences will be finite given Kripke's theorem, Theorem 2.13: namely, those sequences whose generating set does not contain an invertible formula. So we know that a proof search tree, $pst(\alpha)$, for $\alpha$, based on $L_5$, will have only finitely many multisets not containing invertible formulas. We now show that $pst(\alpha)$ contains only finitely many multisets that have invertible formulas as members. Because of the (A) conditions on the rules $Pv''$ and $Po''$ in $L_5$, and given the
discussion surrounding Theorem 2.30 which provided our normalization of
L3-proofs, we know that if $\alpha$ is L5-provable, then there will be a normalized
L5-proof of $\alpha$. As a result, in constructing $pst(\alpha)$, where $\alpha$ contains at least one
invertible formula, say $A$, we need only generate the subtree of $pst(\alpha)$ that has $A$ as
principal in $\alpha$ (i.e. we have an irrevocable strategy for trying to prove multisets
which contain invertible formulas). But our invertible rules, $P^+$ and $P^&$, take
premisses which are properly simpler than the conclusions they give rise to (i.e. the
degree of the premisses is strictly less than the degree of the conclusion). So given
finite $\alpha$, the process of taking invertible formulas as principal must terminate,
leaving us with a layer of multisets that contain no invertible formulas. In searching
for premisses for multisets in $pst(\alpha)$ that do not contain invertible formulas, we
may release invertible components, but again, we use normalization, and this
process must terminate again in a layer of multisets not containing invertible
formulas. But Kripke's theorem assures us that there can only be finitely many
multisets in $pst(\alpha)$ not containing invertible formulas, and so we have finite $pst(\alpha)$,
for any $\alpha$. Q.E.D.

Theorems 2.33 and 2.36 satisfy the goals of this section: we have a Gentzen
formulation L5 which is sound and complete with respect to LR, and which is
decidable. Moreover, we have satisfied our aim of achieving tighter controls on the
contraction principle than what Lk manages. Indeed, these controls would seem to
be maximal. The effect of contraction is restricted to the principal constituent of an
v-introduction rule, and to the principal and parametric constituents of a o-
introduction rule, and tightly governed by the number of occurrences of a
constituent in a multiset, and as the following examples illustrate, there are
LR-provable formulas that require at least this amount of contraction. For
example, the LR-theorem

$$(pvqv(\sim p&\sim q)))$$

requires the contraction built into (B)(b) of $Pv''$. Indeed, its only L5-proof is
with the *'d step instancing (B)(b). The fact that contraction is (sometimes)
required on parametric members of an application of the Po" rule (i.e. that
condition (B)(a) is required) can be seen by inspecting the proof search tree for the
contraction axiom A4 of LR. That contraction is (sometimes) required on the
principal constituent of an application of Po" rule (i.e. that conditions (C)(a) and
(C)(b) are required) follows by inspecting the proof search tree for WF, an LR
theorem the reader will recall from Chapter 1. In nrm-form, WF is

\[
WF'. \quad (\text{po}(\text{qo}q) + (\text{p} + (\text{p} + (\text{q} + q)))
\]

and an L5-proof of it appears as

\[
\quad \text{p} \quad \text{q} \quad \text{q} \quad \text{p} \quad \text{q} \\
\quad \text{p} \quad \text{q} \quad \text{q} \quad \text{p} \quad \text{q} \\
\quad \text{p} \quad \text{p} \quad \text{po}(\text{qo}q) \quad \text{qo}q \quad \text{q} \quad \text{p} \quad \text{q} \\
\quad \text{po}(\text{qo}q) \quad \text{p} \quad \text{p} \quad \text{q} \quad \text{q} \\
\uparrow \\
WF'
\]

Any reader interested in trying to improve the controls that L5 places on the
contraction principle should note these three cases, in particular.
2.2.5. Computational Complexity Revisited.

Although we have limited the use of contraction, and have been able to restrict our appeal to Curry's Lemma as a result, our proof of decidability still embraces both, and so the ingredients for our concern regarding Kripke's conjecture can still be found in the pot. But our successes have been significant enough to incline us to some, albeit mild, degree of doubt concerning that conjecture. For L5, while perhaps being rather unwieldy for human theorem provers, is considerably more computational restrained than Lk, and ideal as a basis for Kripke. Given that we still lack a definitive answer to Kripke's conjecture concerning the computational complexity of theorem-proving in LR, it is rather hard to quantify our successes. Later we will compare how Kripke, based alternatively on Lk and L5, measures up. But these measures apply to performances on particular formulas, or restricted classes of formulas, and have at best a loose kind of statistical significance. Despite our difficulties in assessing any global improvement in the move from Lk to L5, we can assess local improvement, in the sense that we can recalculate the McRobbie function for giving the number of premisses or premiss-pairs for a multiset \( \beta \) in \( \text{pst}(\alpha) \).

If a multiset \( \beta \) contains an invertible formula, then there will be exactly one premiss or premiss-pair, depending on whether \( \beta \) was derived using \( P^+ \) or \( P^- \), respectively, from which \( \beta \) can be derived. If \( \beta \) contains no invertible formulas, and so was derived using \( P^0'' \) or \( P^v'' \), then the number of premisses or premiss-pairs will very much depend on how many times each generator of \( \beta \) occurs in \( \beta \). Consider the rule \( P^0'' \). In Lk, an application of \( P^0 \) has the number of its premiss-pairs bounded above by \( 4x3^n \), where \( n \) is the number of parametric members of \( \beta \) (i.e. \( \text{crd}(\beta)-1 \)). This bound will only occur in L5, for an application of \( P^0'' \), when every generator of \( \beta \) occurs exactly once in \( \beta \).

In general, the bound can be calculated as follows. Let \([\text{AoB}, \alpha]\) be the multiset, with AoB principal, and we are interested in the number of pairs of premiss multisets from which we could have derived \([\text{AoB}, \alpha]\) using \( P^0'' \). The components of
the fusion formula introduced by $Po''$ will have to belong to each pair of premisses - one to every left-hand premiss, and the other to every right-hand premiss - so they need not enter into the calculation. We need only be concerned with how to divide up $\alpha$, the parametric members. Of course, any selection, $\beta$, of members of $\alpha$, including the null selection, could have come from, say, the left premiss, provided the balance of $\alpha$, say $\gamma$, (perhaps with some copies of members of $\beta$, where permitted by the conditions on the rule), is assigned as parametric in the right premiss. So we determine choices of $\beta$ first, and this amounts to calculating the power multiset of $\alpha$. Represent $\alpha$ as

$$[[C_1, \ldots, C_k]],$$

where $C_{ni}$ stands for the $n_i$ instances of generator $C_i$ of $\alpha$. Now there are $(n_i+1)$ ways of selecting up to (and possibly all and possibly none) of the $n_i C_i$'s, so there will be

$$P(\alpha) = \sum_{i=1}^{k} (n_i+1)$$

ways of selecting from $\alpha$, and any such selection will be a possible set of parametric members, $\beta$, for the left-hand premiss $[[A,\beta]]$. For example, the power multiset of $[[A,A,B]]$ is

$$\{[\ ], [A], [A,A], [B], [A,B], [A,A,B]\}$$

and so $P([[A,A,B]])=6$; there are six possible left-premisses. Having fixed choices for the left premiss, we now make choices for the right-side premiss, $[[B,\gamma]]$. Now, if for a given $C_i$ its $n_i$ is greater than 1, condition (B)(b) of $Po''$ informs us that where we selected $k$ of the $n_i C_i$'s to go into $\beta$, exactly the remaining $n_i-k C_i$'s (if any) will have to go into $\gamma$. So the only way we can get a particular left-premiss paired with more than one right-premiss is for some $n_i$ to equal 1, and the $C_i$ in question to already be in $\beta$. Condition (B)(a) will then permit us to either put $C_i$ in $\gamma$, or omit $C_i$ from $\gamma$. But for $n_i=1$, the number of ways of handling $C_i$ into a premiss-pair is just that of $P_{ko}$ in $L_k$; i.e., three: into the left, the right, or both. Condition (C)(a) is the only one we have still to account for. If the principal constituent, $AoB$, does not occur in $\alpha$, then we have the option of taking it into $\beta$, or $\gamma$, or both, or neither - that is, four possibilities. So our function, $Cnt$, for determining the number of distinct premiss-pairs for an application of $Po''$ to give $[[AoB,\alpha]]$ is
\[ Cnt(AoB;\alpha) = P(\delta) \times 3^k \times \min(4, (4 \times c(AoB;\alpha))) \]

where \( \delta \) contains all \( C \) such that \( c(C;\alpha) < 1 \) and
\[ k = \text{the number of generators } D \text{ in } \alpha \text{ with } c(D;\alpha) = 1 \]

Comparison of the premiss-pair functions for \( Pk_0 \) and \( Po^n \), on the following sample multisets, will give the reader some feel for what we have achieved:

<table>
<thead>
<tr>
<th>Multiset</th>
<th>Pko-pairs</th>
<th>Distinct</th>
<th>Po^n-pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>([AoB,C,D,E,F,G])</td>
<td>972</td>
<td>972</td>
<td>972</td>
</tr>
<tr>
<td>([AoB,C,C,C,C,C])</td>
<td>972</td>
<td>84</td>
<td>24</td>
</tr>
<tr>
<td>([AoB,AoB,C,C,C])</td>
<td>324</td>
<td>80</td>
<td>12</td>
</tr>
</tbody>
</table>

Note especially that it is not only the decreased number of \( Po^n \) pairs that is significant. Also significant is the fact that, on average, the \( Po^n \)-pairs will be less complex than the \( Pko \)-pairs, in the sense that if we recursively apply the respective premiss-pair counting functions to the \( Pko \)-pairs and the \( Po^n \)-pairs, sum these counts, and divide through by the number of initial pairs, we will get a considerably smaller average on the \( Po^n \)-pairs. We direct the reader's attention to Table 2-7 at the end of this section for more general results along these lines.

Given a particular multiset, \( \alpha \), and a particular choice of principal constituent, \( A \), we shall use this premiss-pair counting function as a guide to the 'cost' of searching for a proof of \( \alpha \) with \( A \) as principal. Indeed, in Chapter 4 we will consider other factors that might also contribute to this 'cost'. We will then be in a position to intelligently guide our search through \( pst(\alpha) \).
### Table 2-7: Premiss Pairs for \( P_0'' / P_{ko} \)

<table>
<thead>
<tr>
<th>No. copies of</th>
<th>No. copies of principal constituent</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, B, C</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No. of copies of</th>
<th>( P_0'' )</th>
<th>( P_{ko} )</th>
<th>( P_0'' )</th>
<th>( P_{ko} )</th>
<th>( P_0'' )</th>
<th>( P_{ko} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>1 0 0</td>
<td>12</td>
<td>12</td>
<td>9</td>
<td>24</td>
<td>9</td>
<td>39</td>
</tr>
<tr>
<td>2 0 0</td>
<td>12</td>
<td>24</td>
<td>9</td>
<td>48</td>
<td>9</td>
<td>78</td>
</tr>
<tr>
<td>3 0 0</td>
<td>16</td>
<td>40</td>
<td>12</td>
<td>80</td>
<td>12</td>
<td>130</td>
</tr>
<tr>
<td>1 1 0</td>
<td>36</td>
<td>36</td>
<td>27</td>
<td>72</td>
<td>27</td>
<td>117</td>
</tr>
<tr>
<td>2 1 0</td>
<td>36</td>
<td>72</td>
<td>27</td>
<td>144</td>
<td>27</td>
<td>234</td>
</tr>
<tr>
<td>3 1 0</td>
<td>48</td>
<td>120</td>
<td>36</td>
<td>240</td>
<td>36</td>
<td>390</td>
</tr>
<tr>
<td>2 2 0</td>
<td>36</td>
<td>144</td>
<td>27</td>
<td>288</td>
<td>27</td>
<td>468</td>
</tr>
<tr>
<td>3 2 0</td>
<td>48</td>
<td>240</td>
<td>36</td>
<td>480</td>
<td>36</td>
<td>780</td>
</tr>
<tr>
<td>3 3 0</td>
<td>64</td>
<td>400</td>
<td>48</td>
<td>800</td>
<td>48</td>
<td>1300</td>
</tr>
<tr>
<td>1 1 1</td>
<td>108</td>
<td>108</td>
<td>81</td>
<td>216</td>
<td>81</td>
<td>351</td>
</tr>
<tr>
<td>2 1 1</td>
<td>108</td>
<td>216</td>
<td>81</td>
<td>432</td>
<td>81</td>
<td>702</td>
</tr>
<tr>
<td>3 1 1</td>
<td>144</td>
<td>360</td>
<td>108</td>
<td>720</td>
<td>108</td>
<td>1170</td>
</tr>
<tr>
<td>2 2 1</td>
<td>108</td>
<td>432</td>
<td>81</td>
<td>864</td>
<td>81</td>
<td>1404</td>
</tr>
<tr>
<td>3 2 1</td>
<td>144</td>
<td>720</td>
<td>108</td>
<td>1440</td>
<td>108</td>
<td>2340</td>
</tr>
<tr>
<td>3 3 1</td>
<td>192</td>
<td>1200</td>
<td>144</td>
<td>2400</td>
<td>144</td>
<td>3900</td>
</tr>
<tr>
<td>2 2 2</td>
<td>108</td>
<td>864</td>
<td>81</td>
<td>1728</td>
<td>81</td>
<td>2808</td>
</tr>
<tr>
<td>3 2 2</td>
<td>144</td>
<td>1440</td>
<td>108</td>
<td>2880</td>
<td>108</td>
<td>4680</td>
</tr>
<tr>
<td>3 3 2</td>
<td>192</td>
<td>2400</td>
<td>144</td>
<td>4800</td>
<td>144</td>
<td>7800</td>
</tr>
<tr>
<td>3 3 3</td>
<td>256</td>
<td>4000</td>
<td>192</td>
<td>8000</td>
<td>192</td>
<td>13000</td>
</tr>
</tbody>
</table>

Note that A, B and C are distinct literals, and the principal constituent is a fusion of literals. Note also that the figures given for \( P_{ko} \) are for distinct premiss-pairs, rather than the inflated count given by the McRobbie function.
2.3. Properties of Provably Multisets

In the last section we concentrated on the local properties of rules, proofs, and proof search trees, culminating in the formulation L5 of LR. In the first part of this section we continue this approach, discussing further ways of normal-forming inputs to Kripke, and the use of derived rules and axioms. In the latter part of this section we discuss various global properties of provable multisets. A global property of a provable multiset, \( \alpha \), is a property which \( \alpha \) must necessarily have if there is to be a proof of \( \alpha \). Importantly, that \( \alpha \) has a particular global property is determined by inspecting \( \alpha \), outside the context of a proof search tree. In the context of tree-based searches, inspecting nodes of the tree for compliance with global properties will amount to what is termed 'node pruning'. Thus, if a multiset \( \beta \) has the global properties we require of it, we continue the construction of \( \text{pst}(\alpha) \), on \( \beta \), in search of a proof, and otherwise, we abort construction of the branch containing \( \beta \).

2.3.1. Normal-Forms Revisited.

With L2 we saw the utility of considering negation-normal-forms of formulas input to Kripke. This move implied a change in our underlying propositional language, and our stock of primitive connective rules, and so required an argument that such a move was admissible from the local perspective of a proof. The normal-forming we consider now does not imply such a change, and as the transformations of formulas input to Kripke that we consider here all preserve provable equivalence, we may freely avail ourselves of them.

We mentioned earlier that all the active connectives not eliminated in the process of taking nrm-forms, namely \&, v, o, + are provably associative and commutative in LR. That is

\[
\begin{align*}
\text{AS}\&. & \iff ((A&B)&C) \iff (A&(B&C)) & \&-\text{association} \\
\text{C}\&. & \iff (A&B) \iff (B&A) & \&-\text{commutativity}
\end{align*}
\]

and similarly for v, o and +. Thus, the two conjuncts in

\[ F2. \quad ((Ao\sim B)oC) \& (Ao(Co\sim B)) \]
while being syntactically distinct, are logically equivalent. Also recall that the conjunction and disjunction connectives in LR, & and v, are modelled on the lattice operations of meet and join, respectively, and so are provably idempotent and absorbtive in LR. That is

- **ID&.** \( \vdash (A \& A) \Leftrightarrow A \) \hspace{1cm} Idempotence of &
- **IDv.** \( \vdash (A \lor A) \Leftrightarrow A \) \hspace{1cm} Idempotence of v
- **AB.** \( \vdash (A \& (A \lor B)) \Leftrightarrow (A \lor (A \& B)) \Rightarrow A \) \hspace{1cm} Absorbtion

We note, regarding the propositional constants, that in LR

- **It.** \( \vdash t \Rightarrow A \Rightarrow A \)
- **IT.** \( \vdash T \& A \Rightarrow A \)
- **If.** \( \vdash f \lor A \Rightarrow A \)
- **IF.** \( \vdash F \lor A \Rightarrow A \)

and, of course, that

- **VT.** \( \vdash T \lor A \Rightarrow T \)
- **&F.** \( \vdash F \& A \Rightarrow F \)

Trivial syntactical distinctions like these are annoying. Most importantly, they give rise to a formula having more subformulas than we need to be interested in. For example, F2 has eight distinct subformulas, whereas given ASo, Co and ID&, F3. \(((A \& B) \& \neg C)\) is provably equivalent to F2, and has three fewer distinct subformulas. We saw in Section 2.2.5 that the complexity of deciding the provability of some formula input to KRIPKE was, in a loose sense, bounded by the number of distinct subformulas of that input. Decreasing the number of distinct subformulas can thus only help.

Consider the proof search tree with the following initial formula

- **F4.** \( (A \lor ((B \& (\neg (A \& A)) \& (A + (\neg B + C))) \& A) \)

It will be quite large, as F4 is not provable. But by absorbtion F4 is clearly equivalent to just A.

To take account of the associativity and commutativity of \&, v, o and +, and the idempotent and absorbtive properties of & and v, we begin by treating the
connectives as *n-ary connectives* taking a multiset of arguments. Where a
connective occurs outside of multiset brackets, as in &[[A,B,C]], it indicates that all
the members of the multiset are connected by that connective (e.g. as A&B&C).
Thus F4 would be represented as

F6. v[A, &[B, o[¬A, A], +[A, ¬B, C], A]]

That we can consider the arguments of our n-ary connectives to be multiset-
grouped follows from associativity and commutativity. We shall impose a univocal
ordering by placing a particular lexicographical order on the arguments in a
multiset (the details of which need not concern us). Thus F6 will be uniquely
represented as

F7. v[A, &[A, B, o[A, ¬A], +[A, C, ¬B]]]

Any formula so represented will be called an *ordered n-ary formula*. Note that any
nrm-formula will have a unique equivalent ordered n-ary formula. Using
exponential notation for representing members of a multiset, we will generalize the
principles of idempotence and absorption to n-ary form, for finite n, as

<table>
<thead>
<tr>
<th>IDn&amp;</th>
<th>&amp;[A^n, β] ⇔ &amp;[A, β]</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDnv</td>
<td>v[A^n, β] ⇔ v[A, β]</td>
</tr>
<tr>
<td>ABn</td>
<td>&amp;[A, v[A^n, β]] ⇔ v[A, &amp;[A^n, β]] ⇔ A</td>
</tr>
</tbody>
</table>

and likewise for It, IT and the like. It is clear now that by applying ABn to F7 we
get just A, as desired. Note, however, that when applying IDn& or IDnv we may
have to fix our format, due to the occurrence of singleton multisets. For example,
ID&n would take &([A^n]) to &([A]), which is A. The fix is obvious. Any ordered n-ary
formula to which IDn&, IDnv, ABn, ITn and the like, have been exhaustively
applied (in left-to-right form, of course), will be termed a *simplified ordered n-ary
formula*. A formula in n-ary form can naturally be represented equivalently in the
usual binary form; a simplified ordered n-ary formula, transformed into binary
form, will be termed a *simplified formula*. The following lemma is immediate,
from the above discussion

**Lemma 2.37:** Any formula of LR has a provably equivalent, uniquely given,
simplified nrm-formula. Moreover, the subformulas of a simplified nrm-formula are
simplified nrm-formulas.
From now on, we shall consider all input to KRIPKE to be in simplified nrm-form. The process of normal-forming and simplifying will often generate an initial formula that bears little syntactical resemblance to that actually input to KRIPKE. A subsequent proof (if any) output by KRIPKE may appear somewhat mysterious because of this. Once we have a proof of $\text{nrm}(A)$, however, we may use this to directly generate a proof of $A$; again, the details are elementary and need not concern us here.

2.3.2. Derived Axioms and the Ko Rule

Standardly, once the proof-theoretical transformations and reformulations of a logic have been completed one normally shows that $Axp$ can be generalized to include a multiset consisting of any formula and its negation as an axiom. Thus $Axp$ is often generalized to

$$\text{A,}\sim\text{A} \quad AxA$$

where A is a schematic variable ranging over formulas. However, our move to nrm-formulas, and the elimination of rules for negated connectives (from formulation L2 onwards), undermines the point of such a replacement. Negation, after all, has been restricted to propositional variables and constants, and so $AxA$ will only ever have the strength of $Axp$. In formulations like L1 or Lk, however, $AxA$ is not only admissible, but computationally speaking, more useful than $Axp$. As a consequence, when constructing some $pst(\alpha)$ based on Lk and upon reaching a multiset such as

$$\beta = \[(\text{AoB}),\sim(\text{AoB})\]$$

on some branch, one can terminate the construction of that branch in the knowledge that $\beta$ is an instance of an axiom. Although $AxA$ is admissible in L5, its inclusion is, as we have said, pointless. This is somewhat distressing. Translating the members of $\beta$ into nrm-form, we have

$$\beta' = \[(\text{AoB}),\sim(\text{A+~B})\].$$

Although we have no problems regarding the provability of $\beta'$ in L5, on reaching $\beta'$ in some branch of some $pst(\alpha)$ based on L5 we have to go on constructing $pst(\alpha)$. This is a senseless waste of time. The fix here is fortunately easy.
Definition 2.38: If $A$ is an nrm-formula, then its dual is the nrm-formula given by $nrm(\sim A)$.

Lemma 2.39: Any multiset of the form $[A, \text{dual}(A)]$, where $A$ is an nrm-formula, will be L5-provable.

Proof: The proof of this lemma is trivial, and so we leave it to the reader. Q.E.D.

We shall call multisets of the form $[A, \text{dual}(A)]$ derived axioms. The terminology is suggestive of further advances. Consider, for example,

$$\beta'' = [\text{AoB}, \sim A, \sim B].$$

Theorem 2.21 assures us that if $\beta'$ is provable, which it is, then so is $\beta''$. There is no reason why we should not include multisets of the form $\beta''$ as derived axioms, also.

Heeding the full suggestive power of our terminology, if we take the propositional variables $p_1, \ldots, p_n$ of our normal-formed input to Kripke, and derive L5-provable multisets $\alpha_1, \ldots, \alpha_k$ from axioms of the form $[p_1, \sim p_1], \ldots, [p_n, \sim p_n]$ in the normal Gentzen manner in such way that every member of any of the $\alpha_1, \ldots, \alpha_k$ is a subformula of our normal-formed input, then there is no reason why we should not also include the multisets $\alpha_1, \ldots, \alpha_k$ amongst our derived axioms. On encountering one of the $\alpha_1, \ldots, \alpha_k$ in some branch of our $pst(\alpha)$ we can immediately terminate the construction of that branch, in the knowledge that the multiset in question will be provable. The explicit advantage accrued by Kripke deriving some such set of $\alpha_1, \ldots, \alpha_k$ in the usual Gentzen manner, either before setting out on the construction of $pst(\alpha)$ proper or during the construction, is that deriving provable multisets from axioms is a computationally simple problem, at least compared to searching for them within $pst(\alpha)$. Deriving provable multisets cannot replace the process of constructing a proof search tree - at least if we want a guaranteed decision procedure - but it can help, within that general strategy. We note in passing that this combined approach - proof search, with normal Gentzen derivation from axioms - is very reminiscent of what [Nilsson 80] (p.88) terms bidirectional search.
There are, however, several strategic questions left open by our discussion of derived axioms: which of the $\alpha_1, \ldots, \alpha_k$ are really worth remembering, and which could just as easily be re-proved within the proof search strategy proper; how small should $k$ be, considering the overall time and space requirements of Kripke; and so on. As our concern here is still principally one of examining the logical questions concerning automated theorem-proving for LR, we will leave further discussion of these questions until Chapter 4.

We now turn our attention to one final rule that we shall want to admit to LS. Astoundingly, given the scorn we have shown for the structural rule of weakening $K$, our candidate is actually a restricted version of it, namely

**Figure 2-8: The $K_0$ Rule**

\[
\begin{align*}
A, \alpha & \quad K_0 \\
A, A, \alpha & \\
\end{align*}
\]

where either (i) $[t]/\alpha$ (ii) $[B, \neg B]/\alpha$

That this rule is admissible in LS follows from an observation of Meyer's to the effect that the following are each LR-provable formulas:

F8. $(C \rightarrow A \rightarrow B \rightarrow B) \rightarrow (C \rightarrow A \rightarrow A \rightarrow B \rightarrow B)$

F9. $(C \rightarrow A \rightarrow t) \rightarrow (C \rightarrow A \rightarrow A \rightarrow t)$

These equivalences give rise to the following admissible rules:

\[
\begin{align*}
\alpha, A, B, \neg B & \quad \alpha, A, t \\
\alpha, A, A, B, \neg B & \quad \alpha, A, A, t
\end{align*}
\]

which is just our candidate $K_0$, with conditions (i) and (ii). Note that $K$ and $K_0$ differ in essential respects (as they had better, if we are to keep out of trouble). In particular, $K_0$ can only ‘weaken in’ a copy of a formula which already has an instance in the premiss, and even then, only when either of conditions (i) or (ii) apply. We must, however, satisfy ourselves that admitting $K_0$ to LS will not affect the decidability argument.
Theorem 2.40: The system $L_5$ with the rule $Ko$ added, is equivalent to $L_5$, and is decidable.

Proof: The equivalence holds in virtue of F8 and F9. For decidability, note that an application of $Ko$ to some multiset $\beta$ in $pst(\alpha)$ cannot alter the number or nature of the branches at $\beta$, as $Ko$ neither introduces connectives nor alters the membership of the generating set of $\beta$. So our concern reduces to that of showing the finite branch property for $pst(\alpha)$, and so we will be concerned that Lemma 2.35 (restricted Curry’s Lemma) still holds. But Lemma 2.35 follows immediately on Theorem 2.32, and so it will suffice to show that if $\tau$ is a (possibly Ko-augmented) proof of $\alpha=[A,A,\beta]$, then there is a (possibly Ko-augmented) proof $\tau'$ of $\alpha'=[A,\beta]$, in no more steps than $\tau$. The only new case to consider, not handled by Theorem 2.32, is where $\alpha$ is derived using $Ko$ itself. If $A$ was principal, then $\alpha$ followed from $\alpha'$, and thus $\alpha'$ is trivially provable in no more steps than $\alpha$. So assume some member of $\beta$ was principal in $\alpha$, and thus $\alpha=[A,A,C,C,\gamma]$ (i.e. $\beta=[C,C,\gamma]$) and followed from $\alpha_1=[A,A,C,\gamma]$ using $Ko$. But on inductive hypothesis, $\alpha_1'=[A,C,\gamma]$ is provable in no more steps than $\alpha_1$. Using $Ko$ on $\alpha_1'$, we derive $\alpha'=[A,C,C,\gamma]=[A,\beta]$ in no more steps than $\alpha$. Q.E.D.

It can be seen from the proof of Theorem 2.40 that the $Ko$ rule satisfies the antecedent conditions required for an invertibility result along the lines of Corollaries 2.26 and 2.27 for the $P+$ and $P&$ rules. Similarly, Definition 2.29 can be extended appropriately to build the invertibility of $Ko$ into the statement of the rules, and the normalization theorem, Theorem 2.30, can be likewise extended. As a consequence if in the process of constructing $pst(\alpha)$ some multiset $\beta$ satisfies the conditions for being a conclusion of an application of $Ko$, then we can confine our construction on $\beta$ to expanding $\beta$ to find a $Ko$ premiss $\beta'$ for $\beta$. This process can of course iterate as long as $\beta'$ itself satisfies the conditions. (We note in passing that $Kf$ is similarly trivially invertible.) In the context of some $pst(\alpha)$, a continuous sequence of repeated reverse applications of $Ko$ is called a reduction sequence, with the end multiset in the sequence being called the idempotent reduct of the initial multiset in the sequence.
If, in the process of constructing $pst(\alpha)$, we encounter a multiset that could have been derived using any of the invertible rules, $P+$, $P\&$, $Kf$ or $Ko$, with perhaps several candidate principal constituents for each, then selecting any one will of course suffice, and be an irrevocable choice. But just to make for a neat choice, impose an appropriate Markov algorithm for making the selection.

We turn now to discussing global properties of provable multisets.

2.3.3. Positive/Negative Parts and the Rule-of-2

The first global property of interest to us is familiar indeed. That it might be the basis for a node-pruning device was brought to our attention by McRobbie. In the logical literature it is known as either the 'positive-negative parts property' or the 'antecedent-consequent parts property' (see [Anderson and Belnap 75]), and in the context of classical ATP, it amounts to a relevant analogue of the Davis-Putnam pure literal rule (see [Davis and Putnam 60]).

**Definition 2.41:** By the *parts* of some formula $A$, we mean the set of literal subformulas of $A$, counting $t$ and $T$ as $\neg t$ and $\neg T$, respectively, as in Deff and DefF in Section 2.1.1. By the *positive parts*, $pos(A)$, of some formula $A$ we mean the subset of parts of $A$ that are unnegated, and by the *negative parts*, $neg(A)$, of $A$, we mean the subset of parts of $A$ that are negated taking these parts without the negation signs. We extend this terminology to multisets, with $pos(\alpha)$ being the union of all $pos(A)$, $A\in\alpha$, and $neg(\alpha)$ being the union of all $neg(A)$, $A\in\alpha$.

**Definition 2.42:** A multiset $\alpha$ is said to have the *strong positive/negative parts property* iff for all propositional variables, $p$, $p\in neg(\alpha)$ iff $p\in pos(\alpha)$. A multiset $\alpha$ is said to have the *weak positive/negative parts property* iff: (i) $pos(\alpha)\cap neg(\alpha)$ is non-null, or (ii) $T\in pos(\alpha)$.

**Theorem 2.43:** (Anderson and Belnap, Maksimova) If a multiset $\alpha$ contains no
disjunctions, or T or F as subformulas, then α will be L5-provable only if α has the strong positive/negative parts property. Otherwise α will be L5-provable only if α has the weak positive/negative parts property.

**Proof:** The proof can be extracted from [Anderson and Belnap 75] (pp.253-254).

Q.E.D.

Unlike the pure literal rule of Davis and Putnam, a multiset’s failing the strong positive/negative parts property does not license the reduction of the multiset. We are not permitted to delete multiset members that have no matching positive or negative part (as with the pure literal rule), for this involves tacit use of the generalized weakening principal, K. While on the subject of the Davis and Putnam rules for classical ATP, we note that all are inadmissible strategies within relevant ATP, as they all invariably involve tacit use of K. The pure literal rule does have an analogue - the positive/negative parts properties - but the others appear to have no relevant analogues.

It might be thought that the difference between classical and relevant logics, especially the absence of the rule K and related principles - makes for many difficulties and, apart from a superior logical picture of implication, no apparent computational advantages when it comes to actually proving theorems using an ATP system. For example, we noted in Chapter 1 that the absence of K is partly responsible for the blocking of resolution techniques, in Chapter 2.2.5 that W together with the absence of K makes for several complexities even with Gentzen-style techniques, and we have just noted that the absence of K rules out Davis-Putnam deletion strategies within relevant ATP. We now note, though, a global property of relevantly provable multisets which is directly related to the strong and weak positive/negative parts properties and which demonstrates a computational advantage due to the absence of K. The advantage in not having K centers on the sensitive goal-directedness that one would expect from relevance requirements such as variable-sharing.
Definition 2.44: A multiset $\alpha$ is said to have the strict positive/negative parts property iff the set of positive (negative) parts of the complex formulas which are explicit members of $\alpha$ contains the set of negated (unnegated) propositional variables which are explicit members of $\alpha$.

For example, the multiset $\alpha = \{p, q, r, \neg q, \neg r\}$ has the strict positive/negative parts property, whereas $\beta = \{p, q, r, \neg q, \neg r, \neg q\}$ fails to have the property because the explicit negated propositional variable, $\neg q$, has no unnegated mate, $q$, as part of some complex member of $\beta$.

Theorem 2.45: Let $\alpha$ be a multiset containing no propositional constants either explicitly or as subformulas of any member of $\alpha$, and moreover, $\alpha$ is not an axiom. Then $\alpha$ is L5-provable only if $\alpha$ has the strict positive/negative parts property.

Proof: By induction on $n$, the length of proof of $\alpha$. If $n = 1$, then $\alpha$ is an axiom of the form $Axp$, which is ruled out ex hypothesi. So the base case of our induction is $n = 2$, whence $\alpha$ was derived from a premiss or premisses of the form $Axp$. We have four cases to consider, depending on how $\alpha$ was derived.

Case 1: $\alpha$ was derived from $\{p, \neg p\}$ using $P+$, and so is of the form $\{p+\neg p\}$. As the set of explicit propositional variables in $\alpha$ is null, the strict parts property holds trivially.

Case 2: $\alpha$ was derived from $\{p, \neg p\}$ using $Pv''$, and so without loss of generality is of the form $\{p, Av\neg p\}$, which clearly has the strict parts property.

Case 3: $\alpha$ was derived from two premisses each of the form $\{p, \neg p\}$ using $P\&$ and without loss of generality is of the form $\{p, \neg p \& \neg p\}$, for which the theorem holds.

Case 4: $\alpha$ was derived from $\{p, \neg p\}$ and $\{q, \neg q\}$ using $Pd''$, and is of the form $\{p\neg q, \neg p, \neg q\}$; or $\alpha$ was derived from $\{p, \neg p\}$ and $\{p, \neg p\}$ using $Pd''$, and is of the form $\{p\neg p, \neg p\}$. In both general cases the theorem clearly holds.
So assume the theorem holds for all proofs of $\alpha$ of length $< n$; we show it holds for $\alpha$ with length of proof being $n$. On inductive hypothesis, the premiss(es) of $\alpha$ each have the strict parts property. If $\alpha$ was derived using $Ko$, then as $Ko$ does not affect the generators of the premiss but merely the number of repetitions of certain members of the premiss multiset, the strict parts property cannot be violated by an application of $Ko$. If $\alpha$ was derived using a connective rule, then note that the explicit members of $\alpha$ which are negated or unnegated propositional variables must be parametric members of $\alpha$, and cannot be newly introduced into $\alpha$ without already being explicit members of (one of) the premiss(es) (which, of course, need not be the case in the presence of $K$). Whence a little bit of simple set theory and an inspection of the $L5$-rules will show that all of the connective rules must preserve the property. Q.E.D.

In the context of constructing proof search trees, the goal-directing nature of the strict positive/negative parts property becomes clear. Provided the antecedent conditions for the property are satisfied, once a negated or unnegated propositional variable becomes an explicit member of some multiset in some branch of the tree due to the decomposition of some complex formula into simpler parts, then its membership in multisets in this subtree will heavily constrain which rules can be applied to those multisets, until the explicit negated or unnegated variable is isolated in an axiom of the form $Axp$. The strict positive/negative parts property will prove to be an extremely effective node-pruning device in KRIPKE.

The second global property of provable multisets that we will detail in this subsection is called the 'rule-of-2 property'. That such a property holds of provable multisets, and would be useful within our proof search strategy as a node-pruning technique, was first brought to our attention by McRobbie.

**Definition 2.46:** A multiset $\alpha$ is said to have the *rule-of-2 property* iff, if $\alpha$ contains no fusion or $T$ subformula and no explicit $f$'s, then $\alpha$ is provable only if $c(\alpha)$ is less than or equal to 2.
Theorem 2.47: (McRobbie) If \( \alpha \) is \( L5 \)-provable then \( \alpha \) has the rule-of-2 property.

Proof: The proof proceeds by induction on \( n \), the length of proof of \( \alpha \). If \( n=1 \) then \( \alpha \) is an axiom, of the form \( Axt \) or \( Axp \) (note that \( Axt \) is trivial, given Definition 2.46). In either case, \( \text{crd}(\alpha) \leq 2 \). So assume as the inductive hypothesis that the theorem holds for length of proof \( <n \); we show it holds for proofs of \( \alpha \) of length \( n \). So \( \alpha \) was derived via some \( L5 \)-rule. If \( P0'' \) or \( Kf \), the case is again trivial given Definition 2.46. So assume \( \alpha \) was derived using \( Pv'' \), \( P& \) or \( P+ \). But inspection of any of these rules will show that the conclusion of any of them is never of greater cardinality than the cardinality of the premisses, and so because on inductive hypothesis the cardinality of any premiss will be \( \leq 2 \), the conclusion \( \alpha \) will have cardinality \( \leq 2 \). Q.E.D.

The especially nice feature about each of these two global properties is that testing a multiset to see if it has either of them is a very fast procedure, depending solely on the subformulas of formulas in the multiset. The next global property of provable multisets that we consider - the matrix property - is not so computationally neat, but it is extremely powerful, and moreover, variants of it can be used with theorem provers based on any logic whatsoever. For these reasons, we devote the whole of the next chapter to discussing it.
Chapter 3
Algebraic Models

Various researchers in the area of classical ATP, in particular [Reiter 76] and [Sandford 80], have noted the utility of using finite models, satisfying the logic in question, as node-pruning devices, or as devices for guiding heuristic searches. Typically, these models are relational models. Abstract algebras, or matrices, are, as we shall see, just as useful. In Section 1 of this chapter, we give formal clothing to this idea, which we call the matrix property. In so doing we will encounter several problems, both of a logical and computational character, concerning the use of matrices for node-pruning. One problem centers on how to select which matrices to use in KRIPKE, which essentially amounts to the problem of how to assess how faithfully various LR-models in fact model LR. A second problem area centers on the considerable time overheads that evaluating formulas in algebraic models typically involves, and on how these overheads might be reduced to tolerable levels without sacrificing a proportionate degree of logical fidelity. We discuss these and related problems in Sections 3.2 and 3.3.

1 Rasiowa and Sikorski 68] credit [Lukasiewicz and Tarski 30] with calling these algebras 'matrices'. It should be pointed out also that although we arrived independently at the idea of using collections of matrices for node-pruning proof search trees, McRobbie pointed out to us that the outline of a rudimentary form of this idea can be found in [Dummett 77], where, in the context of a discussion of the Gentzen decision procedure for the propositional intuitionistic logic, J, it was suggested that in actually using this decision procedure to construct a proof search tree, the construction could be considerably simplified by using the truth-tables of the classical logic TV, a supersystem of J, for node-pruning.
3.1. LR Models and the Matrix Property

Firstly, we will provide an introduction to various concepts from algebraic model theory that will prove useful in presenting the results of this chapter, and then we will detail the matrix property itself. We will, however, assume some basic familiarity on the reader’s part with the central ideas of algebraic model theory.

The class of algebraic structures that model the logic $R$ is the variety of algebras known as *De Morgan monoids*, with the equational theory of this variety being due essentially to Dunn in [Dunn 66] and [Dunn 75]. De Morgan monoids are structures of the form $<S, \leq, \cdot, o, \sim, t>$ where $S$ is a set, $\leq$ is a partial order of $S$, $\cdot$ is a dyadic operation on $S$ (modelling the connective $\&$), $o$ is a dyadic operation on $S$ (modelling the connective $o$), $\sim$ is a monadic operation on $S$ (modelling the connective $\sim$), and $t \in S$. The actual postulates governing $t$ and the operations are repeated in many places (see, for example, [Anderson and Belnap 75]) and for our purposes it suffices to note that the postulates on the partial order relation are such as to make De Morgan monoids *distributive lattice-ordered structures*. As the postulate for distributivity is independent in the theory of De Morgan monoids, just as A17 is independent in the axiomatization given for $R$ in Section 2.1.1, dropping it from the postulates for De Morgan monoids will give us the variety of algebras modelling the logic $LR$. For want of a better name, we will call these algebras *weak De Morgan monoids*.

**Lemma 3.1:** (Dunn) The class of weak De Morgan monoids is closed under subalgebras, homomorphic images, and subdirect products.

**Proof:** The class is an equational class, and the closure claims follow immediately from Appendix 4 of [Grätzer 79]. See [Dunn 8?] also. Q.E.D.

A *model*, $m$, of $LR$ is a homomorphism from the sentence algebra of $LR$ into a weak De Morgan monoid, with the operation $\sim$ modelling the connective $\sim$, and so on. A formula $A$ is said to *hold* in $m$ iff $t \leq m(A)$. Algebraic structures for
propositional logics can, of course, be regarded as matrices; i.e. structures of the form \( <\mathcal{M}, \mathcal{O}, \mathcal{D}> \), where \( \mathcal{M} \) is some set, \( \mathcal{D} \) a subset of \( \mathcal{M} \) consisting of the so-called designated elements of \( \mathcal{M} \),\(^1\) and \( \mathcal{O} \) a set of operations on \( \mathcal{M} \), with the members of \( \mathcal{O} \) corresponding 1-1 to the connectives in the propositional language \( PL \) underlying \( LR \).

**Definition 3.2:** A matrix model, \( m \), (or 'model', for short), for some formula \( A \) of some propositional language \( PL \), in some matrix \( M \), is then just a mapping \( \theta \) from the propositional variables of \( A \) to members of \( \mathcal{M} \), extended to a homomorphism into \( M \) by interpreting the connectives (and constants) of \( A \) via the corresponding operations (and constants) in \( \mathcal{O} \).\(^2\)

**Definition 3.3:** A formula holds under \( m \) iff \( m(A) \in \mathcal{D} \), and is valid in \( M \) iff it holds in all models (i.e., under all assignments of members of \( \mathcal{M} \) to the variables in \( A \)), and otherwise is said to be invalid in \( M \). If \( A \) is invalid in \( M \), then \( M \) is said to be a counter-model or a refuting-model for \( A \). A matrix is said to satisfy a logic \( L \) iff all \( L \)-theorems are valid in it and \( \mathcal{D} \) is closed under the rules of \( L \).\(^3\) and is said to be characteristic iff exactly the \( L \)-theorems are valid in it. (We have noted previously that \( LR \) has no finite characteristic matrix).

Sometimes we shall want to refer to a particular model \( m \)\(^1\) in matrix \( M \) of some formula \( A \). If we let

\(^1\)Indeed, exactly those members of the corresponding weak De Morgan monoid that are greater than or equal to \( t \) under the partial order.

\(^2\)We will often use the terms 'algebra' and 'matrix' synonymously, although as structures they are somewhat different, with algebras usually omitting mention of some set \( \mathcal{D} \) of designated elements. The conflation of the terms causes no harm here, and indeed proves useful, given that we will repeatedly rely on concepts from algebra, and the field of universal algebra in particular, in developing analogous concepts for matrices.

\(^3\)The requirement that \( \mathcal{D} \) be closed under the rules of \( LR \) amounts to requiring that matrices be strong models (in the sense of [Harrop 65]), a condition which assures the recursive enumerability of \( LR \) matrices.
\[ e^i = m^i(A) \quad \text{(the value A takes under } m^i) \quad \text{and} \]
\[ v^i = <e^i_1, ..., e^i_n> \quad \text{(the underlying assignment to the variables of A)} \]
then we may refer to \( m^i(A) \) via the (not necessarily distinct) \( n \) members of \( \mathcal{M} \) assigned to the \( n \) distinct variables of \( A \), \( v^i \), and the value \( e^i \) that \( A \) takes under that assignment, as the ordered pair

\[ \text{Defm: } <v^i, e^i> \]

Later, we shall want to refer to the set of all models of \( A \) in \( M \), which we shall denote by \( M(A) \). We may sometimes represent \( M(A) \) as a set of ordered pairs of the form \( \text{Defm} \).

We will let \( M, M_1, M_2, ... \) with or without superscripts, range over matrices, and \( m, m_1, m_2, ... \) range over individual models in a matrix. In particular, where \( M_n \) is some matrix, \( M_n \) will be its underlying set of elements, \( O_n \) will be its set of operations and \( D_n \) will be the subset of \( M_n \) containing exactly the designated elements. In describing matrices for \( LR \) we will adopt the following conventions. The extensional operations, for conjunction and disjunction, may always be read off an accompanying partial order, or Hasse, diagram, as meet and join respectively. The operations for fusion and negation will be defined via \( n \)-element ‘truth-tables’, in the usual way, and the tables for implication and fission can be calculated using negation and fusion, as with \( \text{Defo} \) and \( \text{Def+} \) from Section 2.1.1. Note that the constants \( T,F,t \) and \( f \) have corresponding constant elements in a (finite) algebra, with \( T \) always being the greatest element (order-wise), \( F \) least, \( t \) least amongst the designated elements, and \( f \) being the negation of \( t \). We will only mark the location of \( t \) in the Hasse diagram; the designated elements will be those greater than or equal to \( t \) in the order. Note that most of the models we shall examine will not only satisfy \( LR \), but will be De Morgan monoids satisfying \( R \) also.

The principal reason for doing so is simply that we have access to a large number of De Morgan monoids, provided for us by the work of Slaney in [Slaney 80]. Slaney’s work involved, amongst other things, the development of algorithms for the automated generation of algebraic models for relevant logics, and built on the fundamental research of Belnap, Brady, Meyer and Pritchard (all of which is reported in [Slaney 80]).
**Definition 3.4:** Let $S = \{M_1, \ldots, M_n\}$ be some set of matrices satisfying LR, $\beta$ be any multiset, and $\beta^+$ be the formula given by compounding the members of $\beta$ using the fission connective, $+$ (i.e., replace the multiset ",," with a "+"). Then $\beta$ is said to have the *matrix property*, relative to $S$, iff $\beta^+$ is valid in all members of $S$. We extend this terminology to Gentzen-style proofs and say that an L5-proof, $\tau$, of $\alpha$ has the matrix property relative to $S$ iff for all $\beta$ in $\tau$, $\beta$ has the matrix property relative to $S$.

**Theorem 3.5:** If $\alpha$ is L5-provable, then there is an L5-proof, $\tau$, of $\alpha$ that has the matrix property relative to any set of matrices for LR.

**Proof:** The proof of this theorem is straightforward, given the definitions, and so we leave it to the reader. Q.E.D.

In other words, when constructing a proof search tree for some multiset $\alpha$, we need only search it for a proof that has the matrix property. Any path in $pst(\alpha)$ that contains a multiset that fails to have the matrix property can be pruned, at the point of failure. [Reiter 76] notes that the use of general semantic information, like that provided by the matrix property, can not only be used for pruning the search space, but also for often guiding the search. Because of the nature of operational models, and the absence of a direct semantical interpretation of LR proof theory noted in Section 2.1.2, a multiset's failing to have to matrix property does not, at least in the case of LR, inform or direct our search beyond simply pruning nodes. Although the Meyer-Routley semantics does not give us semantical information regarding the the proof search tree proper, it does provide a more direct modelling at the level of formulas and multisets. Thus if $\beta^+$ is *not rejected* by some relational model, $R$, the particular state of $R$ might nonetheless provide information that can be used to constrain various search options; for example, the information might rule out particular members of the multiset from being candidate principal constituents. We are currently investigating the use of Meyer-Routley relational models in Kripke, but at present we have no information.
regarding the practical utility of using them. For those readers familiar with the Meyer-Routley semantics, the problems with using the semantics naturally revolve around the Pasch postulate.

We noted that the positive/negative parts properties, and the rule-of-2 property, are both useful node-pruning devices, and not least because testing multisets in \( \text{pst}(\alpha) \) for these properties is an especially easy and quick procedure. However it is well known that where a matrix has \( 1 \) elements, and \( \alpha^+ \) has, say, \( n \) propositional variables, that there are \( n^p \) basic operations of assigning elements to variables - i.e. there will be \( n^p \) different models for \( A \) - and so testing a multiset in \( \text{pst}(\alpha) \) for the matrix property, although a conceptually easy matter, will computationally be a comparatively slow process. This slowness is disheartening, for among other things early versions of KRIPKE used matrix models to prune nodes in proof search trees and this technique, while being very time consuming, proved especially powerful in that KRIPKE could decide a very wide range of inputs which it previously could not decide even with generous time allowances. But the inordinate amount of time consumed (an amount far too embarrassing to quantify), principally in checking for the matrix property, sometimes for relatively 'simple' inputs, tended to dissuade us from including many matrices in our set \( S \) of matrices to be checked. Initially, we settled on two.

The two matrices in an early version of \( S \) were very small - i.e. they had few elements - but reasonably powerful. The first is a five element matrix known as \text{CHAINS}, and the second is a six element matrix, especially revered amongst relevant logicians, known as the \text{CRYSTAL} lattice.
Figure 3-1: Matrices: CHAIN5 and CRYSTAL

Although CHAIN5 and CRYSTAL are powerful, there are very large classes of LR non-theorems that are valid in these matrices (they are, after all, only very rough finite approximations of LR), and thus initial versions of KRIPKE ran into considerable difficulties in recovering from searching 'bad' paths in \( pst(\alpha) \), for \( \alpha \)'s drawn from these classes.

So in summary, although matrices proved especially powerful as node-pruning devices in KRIPKE, they were time consuming, and given these practical constraints only a few, small matrices could be used. These limitations were somewhat frustrating in that time is of the essence in formal proofs, and even the most powerful matrice will often end up invalidating non-theorems that are valid in some more efficient, but related matrices.
devices in KRIPKE, testing formulas for validity in such matrices was enormously
time consuming, and given these practical constraints only a few, small matrices
could be used. These limitations were somewhat frustrating in that time is of the
essence, the more matrices one can use the closer one will approximate LR, and
larger matrices are in general stronger in the sense that they invalidate more LR
non-theorems. Clearly, what was required was some formal way of comparing finite
models of a logic, in such a way that on the basis of the comparison we could
confidently say that one was in some interesting sense 'better' than another at
invalidating non-theorems of the logic; e.g. it refuted more, or was time-wise more
efficient than most. Rather than rely on intuition to guide the selection of matrices
for use in KRIPKE, a soundly based formal method for comparing matrices would
provide the grounds for selection. In the next section of this chapter we investigate
possible formal bases for comparing matrices of LR.

3.2. Measures of Matrix Strength

Firstly, let us get some idea of our task. We know that all LR theorems are valid
in any LR matrix, but because LR has no finite characteristic matrix, a particular
matrix will often count some LR non-theorem as valid. So, of especial interest to
us is the question of 'how good' a particular matrix is at refuting LR non-theorems.
In answer to our question "What is the LR-refuting 'strength' of the matrix $M$?",
we ideally would like to come up with some measure, and best of all would be a
measure function taking numerical values. For example, "matrix $M$ refutes 3089
LR non-theorems". Such a measure would enable us to not merely compare $M_1$
and $M_2$ and conclude that $M_1$ was better at refuting LR non-theorems, say, but
would enable us to calculate how much better than $M_2$ it was. Thus, our method of
measurement, aside from having a sound theoretical basis, must ideally be one that
is computationally feasible. We actually want to assess how much better one matrix
is over another.

Various ways of comparing the 'strengths' of formal systems, theories or models
have been suggested by several authors, along with impressive applications of these techniques to questions in formal metatheory. See, for example, [Bunder 82] and [Popper 63]. Some analyses of strength are directed at describing the relative 'richness' of systems, or the 'expressive power' of their underlying languages. Other analyses concern themselves with the content and especially the degree of truthfulness or verisimilitude of a theory or formal system. The latter type of analysis is more in keeping with our concerns here. Common to most of these techniques is the fact that at root they rely on systematic structural connections between the theories or models in question. The techniques we propose here will share this foundation.

The specific question of comparisons of algebraic models for logics was first addressed by Kalicki in a series of papers, including [Kalicki 52a] and [Kalicki 54] and culminating in [Kalicki 52b], who developed techniques for determining whether the respective sets of laws or theorems of any two (finite) algebras were equal, distinct, overlapping, or included one in the other. Unfortunately, these results have two shortcomings from our perspective. The first shortcoming is that they do not involve quantifying the strengths of various matrices, but in simply noting the various order-relations mentioned. Evaluating a formula $A$ in a matrix $M$ for $M$-validity has a certain quantifiable cost associated with it, namely the number of models of $A$ in $M$, and in order that we may assess whether that cost is worth it, for given $A$ and $M$, we need some quantified measure of the expected utility of using the $M$ in question. Secondly, and more importantly, Kalicki's comparisons are on the basis of laws and theorems where theoremhood is determined jointly or separately by the lights of the algebras being compared. In other words, given say two matrices $M_1$ and $M_2$, Kalicki's comparisons of their respective strengths are in terms of $M_1$-theoremhood, $M_2$-theoremhood, or $M_1$-and-$M_2$-theoremhood, where the latter notion is just theoremhood in the direct product, $M_\times$, of $M_1$ and $M_2$, with membership of $D_\times$ being determined by joint pointwise designation in the respective factor algebras. Our comparison, however, must be in terms of LR-theoremhood, and thus must be based on the respective
strengths determined by theoremhood in a the Lindenbaum algebra for LR which is infinite. As joint validity in any finite product of finite algebras is not sufficient for LR-theoremhood, these methods are of no direct use to us. Despite this, and the fact that we discovered Kalicki's research too late for it to be a source of the ideas in this chapter, we commend it to the reader as a clear and valuable introduction to the problems addressed and techniques developed here.

To open our quest for a method on a somewhat philosophical note, it might be thought by some that our desire to find effective ways of measuring the strength of some LR matrix $M$, and thus of comparing the strengths of different matrices, runs up against an immediate and obvious problem: namely, the class of LR non-theorems is infinite, and thus, as some matrices may each refute an infinite number of non-theorems, comparison of their respective refuting strengths will be pointless. This in itself does not necessarily pose a problem. To take an example from the classical propositional logic, TV: although the two-element matrix for TV, which we call $2$, refutes an infinite number of logically non-equivalent TV non-theorems, it is also the case that for given $n$, there are only finitely many $n$-variable formulas which are logically non-equivalent TV non-theorems. For example, there are only 3 TV non-equivalent TV non-theorems refuted by $2$: i.e. $p$, $\sim p$ and $p \& \sim p$. Thus comparison of the refuting strength of various TV matrices can take place step-wise, taking choices of $n$ and examining the number of logically distinct $n$-variable TV non-theorems refuted by the respective models.\footnote{We admit that the case of TV is not especially interesting: $2$ happens to be characteristic for TV, and so the motivation for finding ways of comparing TV-matrices just doesn't get off the ground.}

While the problem of infinite magnitudes is easy to step around in the case of TV, it is not so easy to step around in the case of LR. The following LR-theorem

$$\text{Sql.} \quad A \rightarrow AoA$$

called 'square increasingness' in [Meyer 77], and the fact that Sql cannot be strengthened to an equivalence, assures us that there will be infinitely many

\begin{lemma}
Let $\equiv$ be the binary relation such that $a \equiv b$ says that matrice $a$ is

\begin{lemma}
If a formula $A$ is invalid in some LR matrix $M$, then $A$ is invalid in all LR-matrices which contain $M$ (or an isomorphic copy) as a substructure.

\end{lemma}
logically non-equivalent n-variable formulas in the case of LR for each n>0. In other words, the 'free' weak De Morgan monoid on n generators, for any n, will be infinite. We will never totally overcome this obstacle, but in the next subsection, we will investigate the measure of the relative strengths of matrices that taking account of subalgebras gives rise to. In Section 3.2.2 we will return to the matter of trying to get a measure that permits a tightly quantitative comparison of the strengths of matrices.

3.2.1. Relative Measures of Strength

We firstly collect several facts concerning subalgebras.

**Definition 3.6:** Formally, a subalgebra $M_3$ of $M$ is, given Lemma 3.1, an LR matrix $<M_3, O_3, D_3>$ such that $M_3$ is some subset of $M$ closed under the operations of $O$, where $O_3$ and $D_3$ are restricted to $M_3$. A set $M'$, the closure of which generates a matrix $M$, is said to be a generating set for $M$. A generating set $M'$ is said to be minimal iff no proper subset of $M'$ generates $M$, and a minimal generating set $M'$ for $M$ is said to be smallest (although there may be more than one) iff no minimal generating set for $M$ has cardinality less than that of $M'$.

Next we note some rather obvious lemmas concerning subalgebras.

**Lemma 3.7:** Let $R$ be the binary relation such that 'a$R$b' says that 'matrix a is a subalgebra of matrix b', and let $M_{LR}$ be the class of finite LR-matrices. Then $R$ is a partial order on $M_{LR}$.

**Proof:** Straightforward, given the definitions. Q.E.D.

**Lemma 3.8:** If a formula $A$ is invalid in some LR matrix $M$, then $A$ is invalid in all LR-matrices which contain $M$ (or an isomorphic copy) as a subalgebra.

**Proof:** Again straightforward, so we leave it to the reader. Q.E.D.
Lemma 3.9: Let $M$ be an LR matrix, and $A$ some $M$-invalid formula invalidated by a model $m^i = \langle \langle e_1^i, \ldots, e_n^i \rangle, e^i \rangle$. Then $A$ is invalid in the subalgebra generated by $\{e_1^i, \ldots, e_n^i\}$.

Proof: Again trivial, and left to the reader. Q.E.D.

Lemma 3.10: Let $M$ be some LR-matrix, and $M_1, \ldots, M_j$ be smallest minimal generating sets for $M$, each of cardinality $k$. Let $A$ be any formula having a total of $n$ distinct variables (and constants), where $n < k$. Then $A$ is invalid in $M$ iff $A$ is invalid in some proper subalgebra of $M$.

Proof: Let $m(A)$ be a refuting model for $A$ in $M$. Then $m(A)$ assigns at most $n$ distinct members of $M$ to the variables (and constants) of $A$. (Strictly speaking, constants do not get elements of $M$ assigned to them; but it is clear that if $A$ has at most $n$ distinct variables and constants, then the matrix polynomial corresponding to $A$, under $m$, can have at most $n$ distinct members of $M$ featuring in it.) However the closure of any $n$-membered subset of $M$ cannot, ex hypothesi, generate $M$, but rather, just a proper subalgebra and thus $m(A)$ will be a refuting model for $A$ (up to isomorphism) in this proper subalgebra. Q.E.D.

Combining Lemmas 3.7 and 3.8 clearly provides us with a relative measure of strength of LR matrices. We can now conclude that, if $M_1$ is less than or equal to $M_2$ under the subalgebra ordering, then $M_2$ is at least as strong as $M_1$ (i.e., it will refute anything $M_1$ refutes). We cannot, in general, assume that because $M_2$ is larger than $M_1$, where $M_1$ is a proper subalgebra of $M_2$, that $M_2$ will refute more than $M_1$. A trivial counter-example is the direct power $M^2$ of a given matrix $M$: by Lemma 3.1, $M^2$ will be an LR matrix; it will have more elements than $M$, have $M$ as a proper subalgebra (the one generated by all pairs $<e,e>$, $e \in M$), but it will refute exactly the formulas that $M$ refutes. See [Kalicki 52b] for similar examples.

We can, however, characterize appropriate conditions under which it is accurate to say that $M_2$ is exactly as strong as $M_1$. Those conditions are precisely those
implied by Lemma 3.10. For example, take 2 and the following 3-element LR matrix known in the literature as the $RM3$ matrix but which we call 3:

**Figure 3-2: Matrix 3**

\[
\begin{array}{c|c|c|c|c|c|c}
\hline
 & 2 & 1 & 0 & 1 & 2 \\
\hline
1 = t & 0 & 2 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 2 \\
\hline
\end{array}
\]

Now, the minimal generating sets for 3 are \{2,1\} and \{0,1\}, each of which has cardinality 2. If A is a one-variable formula (having no constants), then that variable can only be assigned either value 0, 1 or 2 from 3, and the value of A on interpretation must, by Lemma 3.10, lie within a proper subalgebra of 3 for the given assignment. Indeed, the proper subalgebras of 3 are all isomorphic to 2 (excluding, of course, the trivial one-element subalgebra). As a consequence, 3 will not only refute all LR non-theorems that 2 refutes, given Lemma 3.7, but given our deliberations over Lemma 3.10 here, will refute exactly the 1-variable formulas that 2 refutes.

The metric of the comparison is still rather limited, however. Where matrices are related under the subalgebra partial ordering, we can assess that one is at least as strong as the other, and in appropriate circumstances, we can assess that it is exactly as strong as the other, for certain classes of LR formulas. But for matrices not related via this partial order, we have no way as yet for even making this rather limited comparison. And ideally, we should like to be able to compare any two matrices - in particular, even where $M_1$ does not refute all the LR non-theorems refuted by $M_2$ and conversely, we would still like to be able to conclude something about their relative refuting strengths. Moreover, our investigation of subalgebra orderings has not provided a way of assessing whether or not one matrix is indeed stronger at refuting LR non-theorems than another, and this must be our main concern.
3.2.2. Rigid Measures of Strength

Within the context of a matrix $M$, a formula $A$ of a logic can quite naturally be viewed as the set of its models in $M$; that is, as $M(A)$. $M(A)$ is obviously a function, the domain of which contains the $n$-tuples of assignments of values from $M$ to the $n$ distinct variables of the formula, and the range of which contains the matching values in $M$ that the formula takes on interpretation in $M$ on the assignments in question. Viewed as functions, it is obvious that all $n$-variable formulas will have the same domain, for a given $M$ and more generally for all $M$ with the same number of elements, although distinct $n$-variable formulas will regularly take different ranges. In particular, a formula valid in $M$ will have its range confined to $D$, whereas an $M$-invalid formula will have an undesignated member of $M$ somewhere within its range. Moreover, this view of formulas as functions gives rise to the following familiar but important observation: a finite matrix $M$ can only 'distinguish' a finite number of $n$-variable formulas qua $n$-ary functions. The truth of the observation follows from the fact that, for given $M$ and $n$, the domain of any $n$-ary function will be fixed and finite, and there are obviously only finitely many ways of selecting possible members for the range of any such function, thus there are only finitely many distinct $n$-ary functions definable in $M$. Specifically, where $M$ has $m$ members, the number of distinct $n$-ary functions definable in $M$ is bounded above by

$$m^m$$

That is, there are $m^n$ distinct $n$-tuples of assignments of members of $M$ to the $n$ variables, each of which may take any of the $m$ elements of $M$ as value.

To state the ideas of the last paragraph somewhat more formally, we will need some notation. We will borrow what we need freely from [Lausch and Nobauer 73], along with several of the basic results mentioned there. We will assume some familiarity on the reader's part with basic notions from universal algebra, such as subdirect product, projection and so on - should this assumption make for difficulties, the reader is advised to consult the principal text in the area, [Grätzer
While much of the discussion will apply, therefore, to algebras universally, we will of course be principally interested in finite LR algebras.

**Definition 3.11:** Let $M^n$ be the direct power of $n$ copies of $M$. Then an $n$-ary function of $M$ is a map from $M^n$ to $M$. The operations of $M$ can naturally be defined (pointwise) on the set of these $n$-ary functions, to give an algebra which we will denote as $\mathcal{F}_n(M)$.

**Lemma 3.12:** (Lausch and Nöbauer) $\mathcal{F}_n(M)$ is isomorphic to the direct power of $m^n$ copies of $M$, where $M$ has $m$ elements.

**Proof:** From [Lausch and Nöbauer 73] (p.20). Q.E.D.

**Lemma 3.13:** If $M$ is an LR-matrix, then $\mathcal{F}_n(M)$ is an LR-matrix.

**Proof:** By Lemmas 3.12 and 3.1. Q.E.D.

**Lemma 3.14:** Let $p_0, \ldots, p_{n-1}$ be the $n$ projections\(^1\) of $\mathcal{F}_n(M)$, and let $\mathcal{R}_n(M)$ be the subalgebra of $\mathcal{F}_n(M)$ generated by closing $\{p_0, \ldots, p_{n-1}\}$. Then $\mathcal{R}_n(M)$ will be an LR-matrix, all and only of whose members will be definable $n$-ary functions in $M$ - i.e. each will correspond to some $M(A)$.

**Proof:** The proof of this lemma is straightforward and so we leave it to the reader. Q.E.D.

Note that we will call $\mathcal{R}_n(M)$ the $n$-ary function space on $M$.

**Lemma 3.15:** Let $M$ be an algebra of $m$ elements, and $\nu_1, \ldots, \nu_k$ be the $m^n$ different sets of assignments of members of $M$ to $n$ variables. Let $M_i$ be the

---

\(^1\)I.e. the functions representing each of the $n$ distinct variables
subalgebra of $M$ generated by $v^i$, and $R$ be the direct product of $M_1\ldots M_k$. Then $\mathcal{R}_n(M)$ is a subalgebra of $R$.

**Proof:** The value of an $n$-ary function on some assignment of values to its variables is obviously constrained to being in the subalgebra generated by the set of those values. (Lemma 3.9 is based on similar considerations.) Thus $\mathcal{R}_n(M)$ is clearly a subalgebra of the direct product of the $m^n$ (not necessarily distinct) subalgebras of $M$ so generated.

**Definition 3.16:** If $\mathcal{R}_n(M)$ is isomorphic to $R$, then $M$ will be said to be $n$-subfunctionally complete.

We now introduce some conventions for illustrating functions, and function spaces. The function defined by $(p_1,\ldots,p_n)$ in $\mathcal{B}$, for example, is:

$$f_1. \{ <0,0>, <1,1>, <2,0> \},$$

where the first member of each pair is of course the assignment to $p$, and the second is the value $(p_1,\ldots,p_n)$ takes in $\mathcal{B}$ on that assignment. This is an especially opaque way of describing a function, let alone compounding on it to describe a space of functions. As the domain of any $n$-ary function in some $M$ of cardinality $m$ will be the same as that of any other $n$-ary function definable in $M$, and as it is principally the range of a function that interests us, we will 'hide' the domain of a function, and represent a function by the values it takes in $M$; i.e. as a sequence of length $m^n$ of members of $\mathcal{M}$. In other words we are taking advantage of the isomorphism mentioned in Lemma 3.12. Thus, $f_1$ would be represented as

$$f_1'. \quad 010.$$

We will extend this convention to our representations of $n$-ary function spaces, as sets of such sequences. Thus, for example, the unary function space on $\mathcal{B}$ is

$$\begin{align*}
012 &= p \\
210 &= \sim p \\
010 &= p\&\sim p \\
212 &= pv\sim p
\end{align*}$$

However, this way of describing spaces is often cumbersome. Consider the following 4-element $LR$-model:
which has a unary function space consisting of the following 64 sequences:

As it happens and will often happen, we could describe this set of 4-tuples succinctly using set abstraction; in this case as

\[ \{ <wxyz> | w,z \in \{0,3\} \text{ and } x,y \in \{0,1,2,3\} \} \]

Rather than use \( w,x,y \) and \( z \), we could use small-face integers 0,1,2 and 3, and describe this set of 4-tuples by

\[ 0,3 \in \{0,3\} \text{ and } 1,2 \in \{0,1,2,3\} \]

with the convention that numbers within set brackets are elements of \( M \), and the space. The 2-ary function space on \( DIAMOND \) can thus be described as

\[ 00,03,30,33 \in \{0,3\} \text{ and } 01,11,12,13, etc \in \{0,1,2,3\} \]

and interpreted in the obvious fashion. We also introduce some notation for describing parts of the range of a function. By

\[ <0,2>, <1,3> \in \{<1,0>, <0,3>, <2,2>\} \]

we shall mean the unary space:

1100 0033 2222 1202 1003 0232 2120 2023
with angle brackets, $<>$, outside the context of set brackets representing an *ordered tuple* of indexes, and inside set brackets representing the corresponding values those tuples can take. This machinery should be adequate for our purposes.

Returning at last to our quest for finitude, we see that although there will be infinitely many distinct n-variable formulas (and as we have seen, an infinite number of logically non-equivalent n-variable formulas, in the case of LR), a finite matrix $M$ will partition the class of n-variable formulas into a finite set of n-ary functions, and indeed, into an n-ary function space $R_n(M)$. Our attempts to obtain measures of the ‘strength’ of $M$ can now turn to giving tightly quantitative measures. For by confining our attention to these finite spaces of definable n-ary functions on various matrices, we can in principal count the number of *invalid* n-ary functions - i.e., functions that have at least one undesignated member of $M$ within their range - and count the total number of definable n-ary functions, and compare the counts for different $M$ and varying $n$.

The catch here is, of course, having a method for finding the exact membership of an n-ary function space, and then getting manageable descriptions of n-ary function spaces, so that in practice we can count and compare. The process of closing the n projections, or the functional representations of the n variables, under the operations of $M$, is often far too time-consuming to be a practically viable method for generating function spaces. We note also that LR is in no way functionally regular - it is not functionally complete nor does it have any closely related property,¹, and LR is not even subfunctionally complete (an inspection of the 1-variable function space of the matrix $A6$, given in [Slaney ??], will suffice to verify this claim). However, by building on techniques from universal algebra, the lemmas presented (especially Lemma 3.15 and Definition 3.16) and ideas of [Tokarz 74] and [Tokarz 75] (especially the notion of ‘similarity’ presented there), we have developed a workable method, and the function spaces described or

¹Such as affine completeness; for definitions of this and other related concepts see [Grätzer 79] (p.409).
mentioned below are produced using that method. To pre-empt the discussion somewhat, our latest attempt to compare the refuting strengths of matrices will leave open pivotal logical questions, and so in the final analysis the method will be tangential to our automated theorem-proving enterprise. We will, therefore, neither describe the method nor prove it adequate.

Looking at the unary function space we have provided for DIAMOND above, we see that there are exactly 4 valid functions, given that the designated elements are 2 and 3, out of a total of 64 functions. In other words, the ratio of invalid functions to total number of functions is 60/64 for DIAMOND. We have already seen that in the case of 2 the ratio is 3/4. Let us take such ratios, provisionally, as the \( n \)-variable refuting-strength of a matrix, and express this ratio for \( M \) as \( St_n(M) = x \), where \( x \) is a rational number between 0 and 1 corresponding to the ratio. On this measure, \( St_1(2) = 0.75 \) and \( St_1(DIAMOND) = 0.94 \). DIAMOND would thus seem to be more powerful at refuting 1-variable non-theorems of LR. And indeed it is. The members 0 and 3 of DIAMOND each generate 2 as a subalgebra, and so if a function corresponding to formula A takes only designated members of 2 (i.e., \( T \) or 1) then the function for A in DIAMOND must, by Lemma 3.10, always take the 3 of DIAMOND as value. However, some functions of the form

\[
3xy3
\]

in DIAMOND have \( x \) or \( y \) undesignated - e.g. the one-variable function 3003 defined by the one-variable formula \( (p+p+\sim p+\sim p) \) - and so DIAMOND clearly invalidates some 1-variable formulas valid in 2. All of this is fairly obvious, but we labour the point with this simple example, to illustrate our technique of comparing matrices. So, moving to function spaces, counting, and taking ratios, seems to have given us the right answer in this case: DIAMOND is stronger than 2 at refuting LR non-theorems - that is, it refutes everything that the 2 refutes, given Lemma 3.8, and more besides.
Exhibiting particular matrices, $M_1$ and $M_2$, and comparing various $St_n$'s for them, does not establish however that the method will be sound in general. To establish the soundness of the method we shall have to show that

(i) where $M_1$ is a proper subalgebra of $M_2$, $St(M_1)$ will not be greater than $St(M_2)$.

Presumably, we want such a guarantee, to ensure that our measure is consistent with the weaker, but logically sound, relative measure developed in the previous subsection.

(ii) for reasons similar to those in (i), where $M_1$ is a proper subalgebra of $M_2$ and there is some $n$-variable formula valid in $M_1$ but not in $M_2$, then we would like a guarantee that $St_n(M_2)$ will be strictly greater than $St_n(M_1)$.

(iii) the numbers make sense as measures of strength (e.g. does the 0.75 figure for 2 and the 0.94 figure for DIAMOND mean that, say, 2 has a 75% chance of refuting a formula selected at random, and that DIAMOND has a 94% chance?)

(iv) the method is generalizable to comparing matrices that are not related via the subalgebra partial ordering, $\mathcal{R}$ (i.e. DIAMOND has 2 as a proper subalgebra, whereas because DIAMOND and, say, CRYSTAL are neither proper subalgebras of the other, the figures given by the measure are, strictly speaking, without interpretation as yet);

Requirements (i) and (ii) would, if met, satisfy us that our approximating move from formulas to functions respects the subalgebra ordering, $\mathcal{R}$, of LR-matrices. Unfortunately, (i) and (ii) fail of $St$ measures for LR. The following counter-example was found by the computer. We generated $\mathcal{R}_1(M)$ for each of 684 distributive LR-matrices (i.e. all the R-matrices of size 10 or less) and 667 non-distributive LR-matrices (i.e. all non-distributive LR matrices of size 9 or less), and ordered the $684 + 667 = 1351$ matrices via the subalgebra partial ordering. We then checked to see whether (i) held. Only two counter-examples were found; we present the simplest. Let $M$ be the following LR-matrix:
\( R_1(M) \) contains 4896 unary functions, of which 4480 are invalid, giving \( St_1(M) \) a value of 0.915. \( R_1(M) \) is exactly described by

\[
\begin{align*}
0, 9 &\in \{0, 9\} \text{ and } \\
\langle 1, 2 \rangle &\in \{\langle 1, 2 \rangle, \langle 8, 7 \rangle, \langle 0, 0 \rangle, \langle 2, 2 \rangle, \langle 7, 7 \rangle, \langle 9, 9 \rangle\} \text{ and } \\
\langle 7, 8 \rangle &\in \{\langle 7, 8 \rangle, \langle 2, 1 \rangle, \langle 0, 0 \rangle, \langle 2, 2 \rangle, \langle 7, 7 \rangle, \langle 9, 9 \rangle\}, \text{ and } \\
\text{the definable 4-tuples for } \langle 3, 4, 5, 6 \rangle &\text{ are} \\
3456 &7777 6333 1111 3543 3453 \\
6543 &2222 3666 8888 6456 6546 \\
7888 &3333 8777 7778 6336 6453 \\
2111 &6666 1222 2221 3633 3546 \\
8887 &9999 7887 3336 8778 \\
1112 &0000 2112 6663 1221
\end{align*}
\]

Note also that the only 1-element generating sets for \( M \) are \( \{4\} \) and \( \{5\} \), and thus by Lemma 3.10 any 1-variable formula invalid in some proper subalgebra of \( M \) will give rise to an invalid function in \( R_1(M) \) on co-ordinates 0, 1, 2, 3, 6, 7, 8 or 9. However, the following is a definable 1-variable function in \( M \) (as can be deduced from the above description):

\[ f_2 = 9992112999 \]

This definable function, \( f_2 \), is valid on all co-ordinates but 4 and 5, which take the undesigned value 1, and so there is some formula valid in all proper subalgebras
of $M$ which is not valid in $M$.\footnote{It occurs to us that there is a neat and effective method for finding the formula, or at least a formula, corresponding to a definable function such as $f_2$. There will of course always be at least one such formula. We have not, however, implemented the method and so we have no idea what the formula for $f_2$ actually is.} Thus, on purely relative measures of strength, $M$ must be properly stronger than any of its subalgebras.

Now $M_3$, where $M_3 = \{0, 1, 2, 3, 6, 7, 8, 9\}$, is a subalgebra of $M$; in fact it is generated by \{3\}. $\mathfrak{R}_1(M_3)$ can be exactly described by:

- $0, 9 \in \{0, 9\}$ and
- $<1, 2> \in \{<1, 2>, <8, 7>, <0, 0>, <2, 2>, <7, 7>, <9, 9>\}$ and
- $<7, 8> \in \{<7, 8>, <2, 1>, <0, 0>, <2, 2>, <7, 7>, <9, 9>\}$ and
- the definable tuples for $<3, 6>$ are
  
  36 63 21 00 11 22 33 66 77 88 99 78 87 12

Thus $\mathfrak{R}_1(M_3)$ has 2016 definable unary functions, of which 1856 are invalid in $M_3$. But this gives $St_1(M_3)$ as 0.920, which is larger than the figure for $M$. Thus, $M$ and $M_3$ suffice to subvert requirements (i) and (ii). We will refer to $M$ in future as $KILLER$, and $M_3$ as $BABY-KILLER$.

Now, we can fix our measure function, $St$, such that requirement (i) is always satisfied, simply by insisting that $M$ inherit the largest $St$ of any of its proper subalgebras. Lemma 3.8 would certainly provide total support for such a move. What can we do to fix (ii)?

As we have seen in the case of the function $f_2$ in the $KILLER$ example, Lemma 3.10 provides a necessary and sufficient condition for one matrix to refute a formula not refuted by a proper subalgebra. Let $f$ be some n-ary function in $\mathfrak{S}_n(M)$, such that

(a) $f$ takes only designated members of $M$ on co-ordinates which generate proper subalgebras of $M$, and

(b) $f$ takes an undesignated member of $M$ on at least some co-ordinate.

Then if $f$ is definable and thus in $\mathfrak{S}_n(M)$ then $M$ refutes some n-ary formula not
refuted by a proper subalgebra. We could go on to build this fact about $M$ into its appropriate $St_n$ as follows. Take the ratio of the number of functions in $R_n(M)$ which satisfy (a) and (b) above, over the number that satisfy (a) alone, multiply the inherited $St_n$ by this ratio, and add the result to the inherited $St_n$ to get a final measure.

From a theoretical point of view, we can consider our $St$ method for measuring the refuting strength of $LR$ matrices to be modified, so as to satisfy (i) and (ii), but from a practical point of view, we will continue to calculate $St_n$-measures using the initial method. We noted before presenting the $KILLER$ counter-example that it was one of only two such examples found, and so we may conclude that $St$ measures based on the initial method will nearly always satisfy (i) and (ii) - at least for matrices and $St$ measures that will be considered in the course of practical automated theorem-proving.

We turn now to requirements (iii) and (iv). The intent of (iii) is obvious, and the parenthetical remark there connecting refuting strength with some probability of refuting a random formula, is clearly in the right spirit. If such a probability was correlated to the $St$ measure, then from the point of view of using matrices in KRIPKE, we would be satisfied. Moreover, an answer to (iii) along these lines would provide the sort of interpretation of $St$ values which could enable valid comparisons between matrices not related via the subalgebra partial ordering, and as such, fulfil requirement (iv). Assessing such a connection is problematical, though. 1 If we make some decision as to what a ‘random formula’ is, then we can start taking probabilities. Fascinated by the opportunity to do some empirical research within logic, we bite the bullet, and decree that, for our purposes, a random $n$-variable formula, $A$, will be any formula in some (sizeable) set $E_n$ of $n$-variable formulas such that $A$ is in nrm-form and the choice and order of symbols in $A$ was determined by consulting some random sequence of numbers (thereby passing the buck).

1 For much that same reasons that, say, assessing the truth of Church’s Thesis is problematical.
So, to test the sensibility of the figures provided by the $St$ method, we ‘randomly’ generated various $\mathcal{E}_i$ letting $i$ range from 1 to 3 (although the results here are primarily for $i=1$) and evaluated the validity of their members within each of the 1351 matrices mentioned above. Then for each $M$ and each $\mathcal{E}_i$, we took the ratio of the number of $\mathcal{E}_i$ formulas invalidated by $M$ over the number of $\mathcal{E}_i$ formulas, and thus empirically fixed measures of the $i$-variable refuting-strength of each $M$. We then compared these measures with the various $St_i(M)$. To illustrate, on the sample matrices used thus far in the text the following figures were obtained:

<table>
<thead>
<tr>
<th>Matrix</th>
<th>$St_i(M)$</th>
<th>$\mathcal{E}_i(M)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.750</td>
<td>0.654</td>
</tr>
<tr>
<td>3</td>
<td>0.750</td>
<td>0.654</td>
</tr>
<tr>
<td>DIAMOND</td>
<td>0.937</td>
<td>0.767</td>
</tr>
<tr>
<td>CHAIN5</td>
<td>0.887</td>
<td>0.748</td>
</tr>
<tr>
<td>CRYSTAL</td>
<td>0.859</td>
<td>0.745</td>
</tr>
<tr>
<td>BABY-KILLER</td>
<td>0.920</td>
<td>0.767</td>
</tr>
<tr>
<td>KILLER</td>
<td>0.915</td>
<td>0.767</td>
</tr>
</tbody>
</table>

As the table shows, the $\mathcal{E}_1$ figures are somewhat skewed, in the direction of being lower than what the $St$ measures would have us expect. Secondly, the $St$ figures make finer-grained distinctions than those licensed by our observations of random behaviour, and in particular, the $St$ values distribute over a wider range than do the $\mathcal{E}_1$ values. Both these problems are reflected in all the observed data. The important question, though, concerns the correlation between $St$ and $\mathcal{E}$ figures for matrices not related via the subalgebra partial ordering, $\mathcal{R}$. To check this, we took all pairs, $<M_1, M_2>$, of matrices not related via $\mathcal{R}$, and observed whether, if $St_i(M_1) \leq St_i(M_2)$ then $\mathcal{E}_1(M_1) \leq \mathcal{E}_1(M_2)$, and conversely. The correlation was exact.
What happens as we vary $i$, though? In brief, the degree of correlation seems to decay rapidly.¹ We will not present the (rather lengthy) tables of comparisons, correlations, and the like that our experiment generated - suffice to say that, for one reason or another, we concluded that something was wrong! Thus, at this point in our empirical foray into logic, we must assume either that

(a) the numbers given by $St$ measures do not make independent sense
(b) that $St$ style measures do not provide a good way of comparing matrices unrelated by the partial order (if indeed there is any good way)
(c) that these discrepancies between expected and observed values are the product of a bad sample (i.e. that our notion of 'random formula' is inadequate), or
(d) that our statistical experiment is a touch too insensitive to display the real and close relations between $St$ measures and our observed $E$ values.

Conclusions (a) and (b) are far too skeptical. Regarding (d), one could try more sophisticated statistical assessments than brute correlations,² but we leave this option to the interested reader to explore. By process of elimination, we are left with (c) as the likely explanation of the absence of correlation. We guessed at various alternative notions of 'random', and repeated the experiment with just as little success, although each run produced different pictures concerning the correlation of $St$ values to $E$ values for particular matrices (which is our principal reason for concluding that we have no idea of what a random formula is at all).

Excuses aside, we thus have no reason to believe that the theory for assessing the refuting strengths of $LR$ matrices given in this subsection will be of much

---

¹We say 'seems' here precisely because we are not sure. Evaluating $i$-variable formulas takes time exponential in $i$, so as $i$ grew we decreased the number of formulas in $E_i$ just so that we could actually evaluate them and get a value for $E_i$. Our doubts, then, concern whether our samples for varying $i$ are uniformly representative.

²For example, maybe matrices which are in some sense 'close' though unrelated in the partial order - e.g. *CHAINS* and *DIAMOND* - are 'as close' when it comes to observed refuting strength.
theoretical use. On the other hand, we would be unhappy if the selection of matrices for use in our ATP systems was to remain a matter of hunches - informed, inspired or otherwise. As a theory of what ought to be the refuting strength of various models, the St measures make a certain amount of sense and go a considerable way to suggesting the kind of theory required. We are, though, still left to contend with the practical problem of selecting matrices for use in Kripke. In the next section, we present our choices, and discuss the hunches that inspired them.

3.3. Testing for the Matrix Property

Firstly, let us reassure the reader that the efforts and angst of the last section were not wasted when it came to the practical matter of determining selection criteria for matrices to use in Kripke. Of the several criteria for determining the membership of our set $S$ of matrices, the initial criterion was a consistently high $St_i$ value, for varying $i$, coupled with a relatively low matrix size.

The other criteria are specific to testing formulas for the matrix property within the context of proof search trees.

Certain matrices are especially sensitive to invalidating certain classes of LR non-theorems, although the $St_i$ values they take indicate that they are not generally useful. In particular, although most non-distributive weak De Morgan monoids have low $St_i$ values compared with distributive weak De Morgan monoids (i.e. De Morgan monoids) of the same size, LR especially prohibits distribution, and so $S$ should and does contain several non-distributive matrices. In similar vein, it is well-known that relevant logics are deductively sensitive to repetitions of premisses,¹ and so $S$ contains several matrices which are particularly attuned to invalidating LR (and R) non-theorems of the form

¹Which is part of the reason for collecting formulas into multisets rather than sets, and partly due to the square-increasingness principle, $S_q$, and the fact that the intensional connectives, $o$ and $+$, thereby differ from their extensional mates, $&$ and $v$, in not being $n$-idempotent for any $n$. (A connective is $n$-idempotent just in case the formula got by connecting $n$ A's with this connective is provably equivalent to any formula connected in the same way containing more than $n$ A's. Thus, $&$ and $v$ are 1-idempotent.)
It was something of a shock to discover, however, that beyond a certain depth in $\text{pst}(\alpha)$ the membership of $S$ did not matter - if $\beta$ failed to have the matrix property, then it was predominantly because $\beta^+$ was invalid in the $2$ - although the depth naturally varied with different $\alpha$. Part of the reason presumably is that the process of expanding multisets in the early part of $\text{pst}(\alpha)$, in combination with various preferred search strategems and checks for the global and local properties of provable multisets, leads eventually to a 'survival of the fittest' state-of-affairs.\footnote{Which prompts the Darwinian conjecture that theorems are multisets that have come down from the trees.} No doubt the primary reason though is that at a certain depth the complexity of multisets in a branch decreases to the point that, if the multiset is unprovable, it is blatantly unprovable, and thus comparatively unsubtle checks such as the positive/negative parts check, or $S=\{2\}$, will suffice to show it unprovable. The depth at which this occurs is not uniform throughout $\text{pst}(\alpha)$, but varies from branch to branch depending on the nature of the multisets in the branch. Let us call such points in a branch matrix thresholds.

More generally, beyond a certain point testing $\beta^+$'s against certain $M$ will not pay its way, but the utility of testing against other $M$ will increase, at least up until a matrix threshold. In particular, the reader will have noticed that the premiss for an application of the $P^+$ rule is a member of the set of singleton premiss-sets for an application of the $P^+$ rule of L5. Thus, if we cash $\gamma$ disjunction in terms of both disjuncts, as

$$\gamma_1 = \frac{A, B, \beta}{P^+}$$

which is one possible way $P^+$ could have been applied, then the subtree on $\gamma$ is no different than if $\gamma$ had been $[A+B, \beta]$. Upon choosing to expand the $\gamma$ as given this
way, the idempotent disjunction gives way to a non-idempotent (tacit) fission, and as a consequence where \( \gamma \) may well be valid in a matrix which is good at refuting instances of BL above, \( \gamma_1 \) may very well not be. The utility of using Sql-sensitive matrices often increases (up until general matrix threshold) the deeper one gets into \( pst(\alpha) \).

As well, some deductive principles of LR have pre-conditions for their use that a given \( \alpha \) may often not satisfy. For example, it might be the case that no member of \( \alpha \) contains a ‘&’ connective, and so no multiset \( \beta \) in \( pst(\alpha) \) can be rejected by a non-distributive matrix \( M \) on the grounds that \( \beta \) involves distributional principles. Having non-distributive matrices as members of \( S \) in this case is a waste of time. Sometimes, situations like this can be detected by inspecting \( \alpha \); other times, not. Detecting when a given \( M \) is incidental to \( pst(\alpha) \) has to rely more often than not then on regular monitoring of the performance of each \( M \) in \( S \).

Naturally, KRIPKE tries to determine and modify the membership of \( S \) according to the sorts of considerations of the previous few paragraphs. However, no matter what sort of algorithms are employed for this purpose, a considerable degree of guesswork will often be involved, and thus mistakes will occasionally occur. On the one hand, KRIPKE may omit from \( S \) a matrix which in fact would refute a particularly obstinate node in \( pst(\alpha) \), where an obstinate node is, as the name suggests, one that has been around in search space for a considerable time, has had a large degree of KRIPKE’s resources devoted to either closing or rejecting it, but which has remained open. To compensate for mistakes like this, KRIPKE will regularly check obstinate nodes for the matrix property against matrices that for one reason or another were absent from \( S \) when the node was previously tested for the matrix property. Reasons for such absence include biases in the selection criteria, the fact that some candidate matrices are large and thus expensive to evaluate formulas in, and so on. On the other hand, KRIPKE may include in \( S \) matrices which in fact will do little if any work at this stage in the construction of \( pst(\alpha) \). Sooner or later, the dynamic monitoring of the performance of members of
S will detect this, and change the composition of S more or less appropriately. But in the meantime, time gets wasted, and checking for the matrix property has the potential to be a great time-waster. There is little that can be done about this sort of mistake, apart from decreasing the average cost of wasted checks. This involves, of course, decreasing the cost of any check for the matrix property, and thus of optimizing the routines for evaluating a formula in a matrix.

Where formula A has n variables, and matrix M has e elements, there are as we have noted many times e^n models of A - i.e. \( M(A) = \{m_1, \ldots, m_{e^n}\} \). The cost of evaluating A in M will then be e^n times some basic cost of evaluating A under some \( m_i \) which will be determined by the length of A. There are various ways of decreasing the basic cost of evaluating A in some given model \( m_i \). Different normal-formed equivalents of A will have lengths less than that of A - e.g. 'B→C' for '~B+C'. Coding the evaluation routines into assembly language can also give an order of magnitude improvement.\(^1\) But by and large, these improvements only nibble at the general exponential problem.

What we really want to do is to decrease the number of models of A in M that we need look at from \( e^n \) to something more manageable. We have two suggestions as to how this can be achieved.

In testing some formula A for validity, it is patently absurd to check distinct copies of isomorphic algebras; the result will be the same for A in each. Let us then close our set of matrices, S, under the taking of subalgebras, and order S so closed by the subalgebra partial order, \( R \). Lemma 3.9 informs us that if a model, \( m_j \) of A

\(^1\) Nice tricks, involving the use of indirect addressing and index registers, can often be used so effectively that substantial parts of the evaluation routine reside and run in register memory. The tricks involve reducing binary operations, such as \& and +, to dummy unary operations - thus evaluating \((A \& B)\) by firstly evaluating A, then B, then the fusion of these values, gives way to the more combinatorial approach of evaluating A, then applying the 'combinator' \&B to the value of A. Further details of these tricks need not concern us here.
in matrix \( M_j \in S \) generates\(^2\) some algebra \( M_a \in S, a < j \), then we need not consider that model \( m_j \) of \( A \). The reasoning is again elementary. Let us consider our set of matrices \( S \) to be closed and ordered as indicated, and our checking for the matrix property to be carried out taking each matrix in \( S \) in turn as it occurs in the order, checking to see whether \( A \) is valid or not in the matrix in question. Should \( A \) be invalid in some such matrix, stop checking - \( A \) will fail to have the matrix property. If, however, we reach some matrix \( M_j \), and some model \( m_j \) of \( A \) in \( M_j \) where the set of values assigned by the model to the variables of \( A \) generates a matrix isomorphic to one earlier in the order of \( S \), then we know that as we have already checked all models of \( A \) in this earlier matrix, and found no counter-example, this model \( m_j \) of \( A \) cannot invalidate \( A \). We can exclude it from consideration.

It should be noted that it is not necessary to close \( S \) in order to get the effect of the savings of the last paragraph. We did this simply to have a canonical copy of isomorphic subalgebras. Providing we uniquely mark an algebra in \( S \), even one which is a proper subalgebra within the context of some algebra in \( S \), as being the representative of all isomorphic algebras generated by models in members of \( S \), with the addition of a little more book-keeping apparatus we can achieve the same effect. To illustrate, consider \( S \) to consist of one matrix \( M \). Partition the models of \( A \) in \( M \) into sets \( C_1, \ldots, C_h \) according to the subalgebra of \( M \) that each generates - that is, if \( m_i \) and \( m_j \) each generate the same subalgebra of \( M \), counting isomorphic subalgebras as distinct, then place \( m_i \) and \( m_j \) into the same \( C_i \). Now, iterate through all the \( C_i \)'s, deleting those \( C_i \)'s whose members generate a subalgebra isomorphic to one generated within some earlier \( C_i \).

Similarly, it should be noted that the range of models in some matrix or set of matrices, \( S \), is relatively independent of the formulas that gave rise to them. We do not have to have to wait to see what formula \( A \) looks like before determining which models within \( S \) will be redundant - all we need know is the number of distinct

\(^2\)In the sense of Lemma 3.9: i.e. the subalgebra generated by the set of values of \( M_j \) assigned by the model to \( A \).
propositional variables in A, because the set of possible models will be determined by the assignments of values to variables, and the force of Lemma 3.9 requires no more information than this (we assume here that A will be free of logical constants). We do not need to know how a particular n-variable formula A determines its value, $e_i$, in $M_a$ under the model $m^i = \langle \langle e_1^i, ..., e_n^i \rangle, e_1^i \rangle$; Lemma 3.9 assures us that whatever $e_i$ is, it lies within the subalgebra $M_a$ of $M$ generated by $\{e_1^i, ..., e_n^i\}$. Thus given some set $S$ of matrices, we can pre-determine which assignments of values to the n-variables of some formula will give rise to a redundant model of any n-variable formula.

In actual practice, we take $S$ at any time as being closed under the taking of subalgebras, ordered under $\mathcal{R}$, and checking for the matrix property proceeds in that order. Lemma 3.10 then gives us an easy way of calculating which models of some n-variable formula in a matrix $M \in S$ will be redundant, for a given choice of n. Namely, any model based on an assignment $\langle e_1^i, ..., e_n^i \rangle$ where $\{e_1^i, ..., e_n^i\}$ generates a proper subalgebra of $M$. As it is often easy to list the subsets of $M$ that are minimal generating sets for $M$ than to list the subsets that generate proper subalgebras, we take the former path. Thus a model $m^i$ of some n-variable formula in $M \in S$ will be redundant iff the set of values assigned to the variables does not contain as a subset some minimal generating set for $M$.

This method of cutting down on the number of models can induce considerable savings, depending on $S$ and the number of variables in A. Tables 3-2, 3-3 and 3-4 illustrate the levels of savings, for (appropriately closed) $S_i$ ranging over the distributive weak De Morgan monoids of size 2, 3, etc. up to size 10, and with n ranging up to 5 variables. In Table 3-2 we give the actual number of n-variable models within each $S_i$, in Table 3-3 we give the corresponding savings in terms of the number of non-redundant models, and then Table 3-4 gives the percentage of the Table 3-2 counts which are thus redundant.
Table 3-2: Numbers of Models in $n$-Variables for Select $S$

<table>
<thead>
<tr>
<th>Number of variables, $n$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>9</td>
<td>27</td>
<td>81</td>
<td>243</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>64</td>
<td>256</td>
<td>1024</td>
<td>4096</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>75</td>
<td>375</td>
<td>1875</td>
<td>9375</td>
</tr>
<tr>
<td>6</td>
<td>108</td>
<td>648</td>
<td>3888</td>
<td>23328</td>
<td>139968</td>
</tr>
<tr>
<td>7</td>
<td>105</td>
<td>735</td>
<td>5145</td>
<td>36015</td>
<td>252105</td>
</tr>
<tr>
<td>8</td>
<td>688</td>
<td>5504</td>
<td>44032</td>
<td>352256</td>
<td>2818048</td>
</tr>
<tr>
<td>9</td>
<td>855</td>
<td>7695</td>
<td>69255</td>
<td>653295</td>
<td>5609655</td>
</tr>
<tr>
<td>10</td>
<td>4610</td>
<td>46100</td>
<td>461000</td>
<td>4610000</td>
<td>46100000</td>
</tr>
</tbody>
</table>

Table 3-3: Numbers of Non-Redundant Models for Select $n$ and $S$

<table>
<thead>
<tr>
<th>Number of variables, $n$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>4</td>
<td>18</td>
<td>64</td>
<td>210</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>44</td>
<td>216</td>
<td>944</td>
<td>3936</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>13</td>
<td>127</td>
<td>913</td>
<td>5671</td>
</tr>
<tr>
<td>6</td>
<td>16</td>
<td>222</td>
<td>1978</td>
<td>14826</td>
<td>101986</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>132</td>
<td>1502</td>
<td>14544</td>
<td>127478</td>
</tr>
<tr>
<td>8</td>
<td>72</td>
<td>1350</td>
<td>16614</td>
<td>173058</td>
<td>1651422</td>
</tr>
<tr>
<td>9</td>
<td>28</td>
<td>846</td>
<td>14392</td>
<td>192128</td>
<td>2263528</td>
</tr>
<tr>
<td>10</td>
<td>274</td>
<td>6764</td>
<td>112618</td>
<td>1570292</td>
<td>19827634</td>
</tr>
</tbody>
</table>

Table 3-4: Percentages of Redundant Models for Select $n$ and $S$

<table>
<thead>
<tr>
<th>Number of variables, $n$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>55</td>
<td>33</td>
<td>21</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>62</td>
<td>31</td>
<td>15</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>93</td>
<td>82</td>
<td>66</td>
<td>51</td>
<td>39</td>
</tr>
<tr>
<td>6</td>
<td>85</td>
<td>65</td>
<td>49</td>
<td>36</td>
<td>27</td>
</tr>
<tr>
<td>7</td>
<td>92</td>
<td>82</td>
<td>70</td>
<td>59</td>
<td>49</td>
</tr>
<tr>
<td>8</td>
<td>89</td>
<td>75</td>
<td>62</td>
<td>50</td>
<td>41</td>
</tr>
<tr>
<td>9</td>
<td>96</td>
<td>89</td>
<td>79</td>
<td>69</td>
<td>59</td>
</tr>
<tr>
<td>10</td>
<td>94</td>
<td>85</td>
<td>75</td>
<td>65</td>
<td>57</td>
</tr>
</tbody>
</table>
To conclude this chapter, we note a second way of decreasing the number of models to be examined. It simply involves ignoring most models of $A$ in a given $M$ - or put more accurately, it involves the random selection of some fixed number of models from each of the matrices in $S$, with the matrix property for a given formula being determined just with respect to these models. Now although a non-theorem will often be refuted by more than one model in a given $M$, our strategy of avoiding redundant models will make this a less frequent occurrence. So the probability that a randomly selected model of $A$ in $M$ will refute $A$, assuming $A$ is refutable, over the cost of looking for a counter-model anyway, will be much less in our favour than what it might first appear to be. Consequently, this method was not incorporated into KRIPKE.
Chapter 4
Proof Theory to Theorem Prover

In this chapter we will detail the move from the proof-theoretic formulation $L_5$ of $LR$ to an implementation of the decision procedure for $L_5$. Whereas in previous chapters we have been concerned principally with modifying the specification of the proof-theoretic formulation of the logic, in this chapter our approach and our concerns will be explicitly computational. To be exact, a successful move from proof-theory to automated theorem-prover has three main components:
- minimizing the number of basic objects in the search space, simplifying the structure of the objects occupying that space, and searching the space 'sensibly';
- and to date we have dealt with only the first of these components. Of course, we have not finished with minimizing the search space, for the three components mentioned are not independent. In particular, in the case of AND/OR decision trees a 'sensible' search regimen is a way of reducing the size of the space to be searched.

In the first section, we will introduce some terminology for stating algorithms, discuss the basic algorithm underlying KRIPKE and examine some alternatives, especially those that concern search strategems. Section 2 will flesh out these basic descriptions, highlighting the more central aspects of KRIPKE. Throughout, we will often refer to the the runtime statistics contained in the appendices to this thesis.
4.1. The Basic Algorithm

We have been remarkably successful to date in concealing the fact that one of the interests of this dissertation is the proving of theorems via a program. From now on, we come clean, and site our discussions within the context of an appropriate algorithm. Our language for presenting algorithms will be (more or less) standard PASCAL, for which see [Jensen and Wirth 74]. We will, however, often neglect to provide type declarations and so on for various datatypes, procedures and the like, where context makes the meaning clear. Nor shall we worry about providing procedure bodies to theoretically insignificant procedures or to procedures which have an obvious algorithm (e.g. the procedure "Print_multiset"). Calls to user-defined procedures and functions will have their identifiers capitalized. PASCAL devotees may be shocked by the use of the 'goto' instruction, and several other flagrant violations of the PASCAL standard, or programming ideology. As KRIPKE is a program of many thousands of lines of code, our intent can only be provide algorithms, and not necessarily to provide executable code. Finally, we note that comments made within the specification of an algorithm will be bounded by the symbols { and }, and we will continue to use [ and ] to delimit multisets.

The basic aim of our theorem-proving program KRIPKE is to automate our decision procedure for LR via the proof-theoretic formulation L5. Hence, for any multiset \( \alpha \), this program should construct \( pst(\alpha) \), checking at each stage of its construction that \( pst(\alpha) \) retains the Curry property. As well, it should inspect \( pst(\alpha) \) to see if it contains, as a subtree, an L5 proof of \( \alpha \) if one exists, and output such a proof when one occurs and terminate. Of course, if \( pst(\alpha) \) does not contain such a proof then KRIPKE will terminate (modulo realistic time and space constraints) and inform us that \( \alpha \) is LR-unprovable. These are the basic requirements, although we will augment KRIPKE with the other features, such as checks for global properties, which were shown admissible in Chapters 2 and 3.

1Copies of various recensions of the code can be obtained from the author.
4.1.1. The Preferred Strategy

The immediate question we face concerns the order in which various parts of a proof search tree should be constructed. We could, for example, adopt any of the following strategies for constructing \( \text{pst}(\alpha) \):

- **Strategy 1:** This strategy amounts to constructing each potential proof of \( \alpha \) one at a time, and searching for proofs of premisses in a depth-first fashion:
  
  (a) pick a new generator, \( A \), of \( \alpha \) as principal - if there is no new \( A \), then \( \alpha \) is unprovable;
  
  (b) select one new premiss-set, \( \{ \beta_1, \ldots, \beta_n \} \), from which \( \alpha \) may be derived using rule \( Pz \) with \( A \) as principal - if there is no new premiss-set, then goto (a);
  
  (c) construct, in turn, \( \text{pst}(\beta_i) \), \( 1 \leq i \leq n \) - if any \( \beta_i \) is unprovable, then goto (b), else \( \alpha \) is provable, with \( \beta_1 \ldots \beta_n \) as premisses, using rule \( Px \).

- **Strategy 2:** as for Strategy 1, but change ‘in turn’ in (c) to ‘in tandem’.

  Although we are still only looking at each potential proof of \( \alpha \) one at a time, our search for proofs of premisses is now breadth-first. For an interesting discussion of breadth-first techniques, see [Siklossy, Rich and Marinov 73].

- **Strategy 3:** as for Strategy 1 or 2, but in (b) select all premiss-sets from which \( \alpha \) may be derived using rule \( Pz \), and perform the (c)-stage construction for each such premiss-set in tandem. If any (c)-stage construction provides proofs of the multisets in its premiss-set, then we have a proof of \( \alpha \), and we abandon the other (c)-stage constructions.

  This strategy is an extended breadth-first approach.

- **Strategy 4:** as for Strategy 3, but in (a) we select all generators of \( \alpha \), and run (b)-stage constructions in tandem for each. This is a global construction strategy, in that it examines all potential proofs in parallel.
All these strategies, including the simple-minded depth-first approach of Strategy 1, are complete in virtue of the decidability argument. We can mix and match various components of these and other strategies, but we will discuss Strategy 1 to start with - the simple depth-first construction of $\text{pst}(\alpha)$. For the moment, we will also assume that our primary data structures - binary trees, the nodes of which contain appropriate control information and multisets of formulas - are well-defined to meet the formal requirements previously imposed on them, and that related substructures, such as the successors and predecessors of a node, the generators of a multiset, and the subformulas of a formula, are likewise well-defined. Although we will provide more details of these data structures in the next section, to do so now would overly complicate the picture. Bearing this in mind, the following algorithm, based on Strategy 1, will suffice as a basic implementation of our decision procedure for $\text{L5}$:

**Algorithm 1**

program KRIPKE;

type
  node_state = (failed, open, closed);
  node_ptr = \*node_info; \{pointer to node information\}
  node_info =
    record \{of node information\}
      left, \{ptr to left premiss\}
      right: node_ptr; \{ptr to right premiss\}
      pc : formula; \{principal constituent\}
      mset : multiset;
      state : node_state;
    end; \{of record\}

function PROOF(var n : node_ptr) : node_state;
var
  applying_rules : boolean;
begin
  \{using the fields of node n\}
  DELETE_f(mset); \{Apply $K_f$ if possible\}
  IDEM_REDUCE(mset); \{Apply $K_O$ if possible\}
  if not CURRY_PROPERTY(n)
    then state :=failed
  else if AXIOM(mset)
then state := closed
else
  if FAILS_GLOBAL_PROPERTIES(mset)
    then state := failed
  else
    repeat {applying rules}
    state := open;
    applying_rules := true;
    if GET_NEXT_PRINCIPAL_CON(mset)
      then case SYMBOL_TYPE(pc) of
          conjunction :
            begin
              GET_CONJ_PREMISES(left, right);
              state := min(PROOF(left), PROOF(right));
            end;
          fission :
            begin
              GET_FISS_PREMISE(left);
              state := PROOF(left);
            end;
          disjunction :
            while applying_rules do
              if GET_DISJ_PREMISE(left)
                then begin
                    if PROOF(left)
                      then begin
                          applying_rules := false;
                          state := closed;
                      end
                end
            else applying_rules := false;
          fusion :
            while applying_rules do
              if GET_FUS_PREMISES(left, right)
                then begin
                    if min(PROOF(left), PROOF(right))
                      = closed
                      then begin
                          applying_rules := false;
                          state := closed
                      end
                end
            else applying_rules := false;
    end {of cases}
  else state := failed;
until state <> open;
proof := state; \{value returned by this call of PROOF\}
end; \{of proof\}

begin \{main program\}
GET_SIMPLIFIED_FORMULA(input);
root.mset := [input].
if PROOF(root)=closed
\{end of program\}
then output(PROOF_OF(root))
else output("Input unprovable");

We note, with respect to Algorithm 1, that

- At any stage, a node can be in one of three states: failed, open or closed. The function PROOF maps nodes to node states, depending on whether it has managed to construct a closed subtree on its argument, n. The states are strictly ordered, so that the min(imum) of a set of node states is that which occurs earlier in the implied order. Thus, min(failed,open)=failed, min(open,closed)=open, and min(x,y)=closed iff x=y=closed. If the arguments to min are function calls, then they will be evaluated in turn, except in the case where the first returns 'failed' and the second argument is thus redundant.

- the rule $Kf$ for introducing the constant, $f$, applied in reverse here in the construction of $pst(\alpha)$, involves simply deleting an instance of $f$ that is an explicit member of the multiset at node n. As $Kf$ is an invertible rule, it can be applied first, and thus repeatedly, to the multiset at node n, until no $f$'s remain explicitly as members of the multiset. We apply $Kf$ before checking for axiomhood, or local or global properties, simply because it produces a multiset that, while being equivalent to the one initially at node n (i.e. provable iff the initial multiset was provable), is simpler. Reversed applications of $Kf$ are managed by the procedure DELETE_f.

- while $L_5$ was not initially formulated with the rule $Ko$, this rule was shown admissible in $L_5$, preserving decidability, in Chapter 2.3.2. Like $Kf$, $Ko$ is an invertible rule, and applied in reverse in the context of
constructing a proof search tree, involves deleting members of the multiset at node \( n \) under certain conditions. The \( Ko \) rule only pertains to multisets that explicitly contain fusion idempotents such as \( t \) or \( (A+\sim A) \). The considerations of the last paragraph apply \textit{mutatis mutandis} to \( Ko \), and reversed applications of \( Ko \) are managed by the procedure \textsc{IDEM\_REDUCE}.

- the generators of a multiset are considered to be ordered so that all invertible formulas, as defined by Definition 2.29, occur before non-invertible formulas. As a consequence, if node \( n \) has an invertible formula as member, it will be selected as the first (and only) candidate principal constituent, \( pc \), in accordance with the constraints of normalized proofs.

- the fission rule, \( P^+ \), is a one-premiss rule, and moreover, there is only one candidate premiss, uniquely determined by the fission formula that is principal. Because \( P^+ \) is invertible, we are assured that if an application of \( P^+ \) fails to produce a closed subtree on node \( n \), then there will be no closed subtree at all. Thus, if a multiset contains an invertible formula, the first and only rule that \textsc{proof} will apply to the multiset in an attempt to produce a closed subtree on node \( n \) will be one of \( P^+ \) or \( P^\& \), as appropriate. Whatever state is returned to such a node will be a definitive state for that node, thus if the call to \textsc{proof}(left) in the \( P^+ \) rule returns failed, we have no interest in applying further rules; we can set the state of node \( n \) immediately.

- the conjunction rule, \( P^\& \), is a two-premiss rule, but like \( P^+ \), has only one candidate premiss-set, and is invertible. The comments of the last paragraph apply \textit{mutatis mutandis} for \( P^\& \) also. The depth-first nature of Algorithm 1 is implicit in the order of recursive calls to \textsc{proof}: the left-hand path will be traversed first, and only after this, together with the assumption that the left-hand path closes, will the right-hand path be examined.

- The disjunction rule, \( P^\lor \), is a single-premiss rule, but as we have noted
elsewhere it has from two to four candidate premiss-sets. The fusion rule, $Pd''$, is a two-premiss rule, with varying numbers of candidate premiss-sets depending on the structure of the multiset at node n. The full conditions of (b) and (c) of Strategy 1 apply to the application of both rules, thus should a particular rule application fail, we cannot set state=failed until we have tried other applications, and other choices of principal constituent.

Generally, we may note that Algorithm 1, as an implementation of Strategy 1, will effect exactly one recursive pass over $pst(\alpha)$, and that no part of $pst(\alpha)$ will be constructed or traversed more than once. In other words, Algorithm 1 has an especially simple control structure, and thus represents an ideal first-up, uncomplicated picture of how to implement a Gentzen-based decision procedure for a logic. The great limitation of Strategy 1 can be seen from examining the algorithms concerned with effecting the $P\&$ and $Pd''$ rules. Consider, as a simple example, the following slice of some $pst(\alpha)$:

**Figure 4-1: AND-wise Failure**

$$
\begin{array}{c}
[A,\beta] & [B,\beta] \\
\lor \\
[A&B,\beta] \\
\uparrow \\
[B,\beta]
\end{array}
$$

and assume that we have just applied the $P\&$ rule to $[A&B,\beta]$ to generate the two premiss nodes displayed. Algorithm 1 commits us to investigating the left premiss, $[A,\beta]$, first - a task which may take quite some time. As it happens, when we come to examine the right premiss, $[B,\beta]$ we see, say, that it fails to preserve the Curry property, and so no matter what the state of the left premiss, that
application of P& fails to produce a closed subtree on node n. We may, of course, alter our algorithm so that checks for the Curry property, and for global properties, are made at the time of selecting a premiss-set rather than within a recursive call to PROOF, but this would not solve the problem. The right side may pass the Curry and global property checks, but nonetheless fail recursively because it fails to have a closed subtree, and do so much more easily than the left side.

This suggests that we should perhaps construct and investigate all paths in $pst(\alpha)$ in parallel - i.e. implement Strategy 4 - so that structural weaknesses in $pst(\alpha)$, like the one just mentioned, come to light as soon as they occur. However, it is clear that Strategy 4 is not the optimal strategy - the optimal strategy is the one that directly delivers of proof of $\alpha$ from $pst(\alpha)$, if there is one. Strategy 4 will often involve a lot of wasted work, where $\alpha$ is provable, although for unprovable $\alpha$ it would seem to be optimal. If $\alpha$ is unprovable, then every potential proof will have to be examined, just to ensure that it is not, after all, an actual proof, and Strategy 4 is strong on detecting structural connections and weaknesses within and between potential proofs. In reality, even in the case of unprovable $\alpha$, Strategy 4 while being time-wise optimal may not be space-wise the best choice. Even with a properly parallel machine, there would be many situations where the number of potential proofs exceeded the number of processors. So Strategy 4, while being a logically appealing approach where we guess that $\alpha$ is in fact unprovable, is computationally unrealistic. Strategy 3 would seem to have similar space problems.

The strategy of our choice forKRIPKE turned out to be a combination of Strategies 1 and 2. As with Strategy 1, our preferred strategy will never have more than one proof-attempt active at any one time and, unlike a version of Strategies 3 and 4, will never abandon or postpone looking at the current proof-attempt to examine another out of sequence. Unlike Strategy 1, however, it will examine the paths of the current proof-attempt in a breadth-first fashion, thereby revealing structural weaknesses in these AND-linked subtrees of the given proof search tree. We see our preferred strategy as a combination of Strategies 1 and 2, in that it does
not effect a simple breadth-first construction of each proof-attempt but rather expands (more or less) of each path of the proof-attempt in an irregular way, depending on an assessment as to the likely cost of expanding the multiset at the tip of an open path.

These assessments of cost are made principally by calculating the number of possible premiss-sets for the given multiset, given by our modified McRobbie function of Chapter 2.2.5. The open tips of the current fragment of $pst(\alpha)$ are expanded in an order and to a degree dependent on these costings. Outside of these costings, there are nodes which Kripke comes to realize as in fact obstinate, and these are penalized also. For example, elsewhere in $pst(\alpha)$ we may have discovered that a multiset $\beta$ at node $n$ failed to have a closed subtree on it, and discovering this may have taken a considerable amount of Kripke’s resources, either space or time. Upon recognizing $\beta$ at node $n'$ in the current fragment of $pst(\alpha)$, we are not permitted to fail $n'$ outright, because failure to have a closed subtree only indicates outright unprovability in the case of the multiset $\alpha$ at the root of $pst(\alpha)$. This point is directly related to our inability, discussed in Chapter 2, to treat proof search trees as graphs. But although node $n'$ could have a closed subtree, it would seem unlikely, and so we avoid expanding such nodes whenever possible.

A common way of implementing breadth-first search strategies is to form a queue of the current open tips of the current fragment of $pst(\alpha)$, and to iterate over this queue expanding cost-wise preferred nodes. Queue-based methods permit iterative rather than recursive control mechanisms, and for various reasons, are preferred by some because of this. We, however, prefer to retain the recursive control mechanisms of Algorithm 1. In order to effect a breadth-first construction of $pst(\alpha)$ we will have to modify Algorithm 1 so that repeated recursive passes over fragments of $pst(\alpha)$ can be made. We will present these modifications shortly, but firstly we note our general reasons for preferring this approach:

- although we have based recensions of Kripke on queue-based control mechanisms, eliminating recursion, the time-savings if any have been
insignificant, and more importantly we have never produced a proof of a formula that was not already provable using our preferred recursive methods. Part of the reason for this is that the time-wise intensive components of KRIPKE lie elsewhere, principally in the application of rules to multisets, and especially in checking for the matrix property.

- queue-based algorithms obscure the natural logic of tree-based decision procedures. To us it seems more natural to construct and examine recursively definable data structures, such as binary trees, using recursive control structures.

- derived axioms, and obstinate nodes, are discovered dynamically, and at the time a multiset $\beta$ at node $n$ is expanded and thus removed from the queue of open tips, it may not be recognized as either. Later on, however, we may discover either that $\beta$, at node $n'$ elsewhere in the tree, is provable and thus a derived axiom, or is obstinate and expansion of the nodes in its subtree should be avoided if possible. But because $\beta$ at node $n$ is no longer in the queue, we have no way taking the appropriate action at node $n$ unless we either locate $n$ directly (say via some hash table of nodes using multisets as the key) or traverse the current fragment of $pst(\alpha)$ checking the nodes in it to see if they are members of either our updated black list of obstinate nodes or our white list of derived axioms. The latter method is easily accommodated within a recursive control strategy.

- recursive control mechanisms can provide space advantages over certain iterative ones. Certain control information and incidental variables that are only temporarily required and need not belong, for example, to a final proof if there is one, can be stored on the calling stack rather than stored as some permanent field of a node.

In vague terms, a scan of the current fragment of $pst(\alpha)$ is a recursive traversal of that fragment. The initial fragment is of course just the multiset containing the appropriately normal-formed input formula. Recursive descent along a given open
path of occurs until a tip is reached, or costing and related information from
previous scans indicates that the particular subtree on a node is not to be
investigated during this scan. If an open tip is reached, it is expanded by some rule
application to produce a set of successor nodes. If the cost of expanding these
successor nodes is tolerable, as in the case of a successor node the multiset of which
contains an invertible formula, then the successor node will be expanded
immediately during this scan in the fashion of Algorithm 1. If a particular rule
application fails, for whatever reason, to produce a closed subtree on a given node,
then other rule applications, or principal constituents will be tried, perhaps
involving backtracking to a predecessor node until an unfailed (but perhaps still
open) path is found. As a scan recursively unwinds along a path, it returns
information about the state of the subtree including data which is used in costing a
proof-attempt, and perhaps marks certain nodes as obstinate or adds others to the
list of derived axioms, and so on. At the conclusion of each recursive scan, a
node_state is returned to the root of \(pst(\alpha)\). When a scan returns a node_state
erother than ‘open’ to the root, scanning terminates and the user is informed either of
the proof of \(\alpha\) or that \(\alpha\) is unprovable. Otherwise, while the root remains open,
iterated recursive scans of (the fragment of) \(pst(\alpha)\) continue. It is important to note
that the costing details never \textit{absolutely} penalize a node, but penalize it relative to
other nodes of \(pst(\alpha)\); consequently some open tip will always be expanded during
any given scan, guaranteeing decidability.

More formally, we can describe our preferred strategy as follows:

\textbf{Algorithm 2}

\begin{verbatim}
program KRIPKE;

{Type and Var declarations as for Algorithm 1;
add local variables and fields to node_info to keep
track of costings and the like.
N.B. We will not detail the costing algorithm here}

function PROOF(var n : node_ptr) : node_state;
\end{verbatim}
begin

{Using the record fields of the node pointed to by n}
case state of
   failed : PROGRAM_ERROR;
      {while a node can take on the node_state
       'failed', it cannot remain in that state from
       one scan to the next, but rather must be
       replaced if possible by an alternative open
       or closed premiss for its parent multiset}
   open   : if TIP_OF_TREE(n)    {are we at a tip?}
            then if CURRENT(n)   {yes - do we
               expand or avoid?}
               then if GET_NEXT_PREMISES(n)
                   {try to expand}
                   then state :=TRY_PROOF(premisses)
                          {call PROOF with each of the
                          premisses, and set the state of
                          node n appropriately}
                   else state :=failed       {can't expand}
                      else begin {avoid - do nothing} end
              else begin {not an open tip} end;
   closed : begin {do nothing} end;
end; {of cases}

{replace failed subtree, if possible}
if state=failed then
   if GET_NEXT_PREMISES(n)    {tried ALL prem-sets?}
      then state :=open;        {no, got another in place
                                 so reset state}
proof :=state; {return value for this call of PROOF}
end; {of proof}

begin {main program}
GET_SIMPLIFIED_FORMULA(input);
root.i.mset :=[input];
root.i.state :=initial;
repeat {recursive scans}
   status :=PROOF(root)
until status<>open;
case status of
   failed   : output("Input Unprovable");
   open     : output(PROOF_OF(root));
   otherwise : PROGRAM_ERROR;
end; {of cases}
end. {of program}
In Appendix A we give runtime statistics for proof searches for formulas from three different sets - Standard, Impset and Asset - with the members of Standard seemingly being less difficult to decide than those of Impset, which in turn would seem less difficult than those of Asset.\(^1\) These statistics would seem to support our preferred strategy.

By examining the percentage of total nodes where an alternative principal constituent had to be selected, or where a node failed due to exhaustion of principal constituents (rather than failing some local or global property), we can get some idea of how often a Strategy 3 or Strategy 4 approach should have been adopted by KRIPKE (but of course wasn’t). If we had have been considering several candidate principal constituents in parallel for these nodes, which in effect amounts to investigating several proof-attempts within $\text{pst}(a)$ at the one time, we may have avoided a considerable amount of work, especially in the case of theorems. This percentage, however, never exceeded 3%.

More importantly, we may note that Strategy 3 or 4 moves cannot help in the case of formulas which are in fact non-theorems of LR, because every proof-attempt will have to be investigated just to ensure that it is not in fact a proof. We note, however, that node-pruning only occurred in 48 of the 104 theorems of Standard, indicating that KRIPKE goes directly to proofs of simple formulas in over half the cases. Unfortunately, for more difficult theorems it seems that some case could be made for incorporating Strategy 3 or 4 moves into KRIPKE. All of the so-called incomplete searches of Asset\(^2\) are in fact for formulas which are LR-provable, and so KRIPKE will eventually find proofs of all of the formulas of Asset although it failed to do so under Algorithm 2 within a tolerable time. As Appendix B shows, some of the incomplete attempts terminated with proof search

\(^{1}\)A complete description of the membership of these sets, and what the statistics actually measure, is given in Appendix A itself.

\(^{2}\)For which, see Appendix A.
trees roughly the size of proofs of proved members of Asset, and so we might perhaps expect Kripke based on Algorithm 2 to find proofs of these formulas given a little more time. However, other searches terminated with rather large spaces still active, which probably indicates that finding a proof in these cases will not simply be a matter of investing, say, another couple of CPU hours in the attempt. Rather, we believe a different strategy is called for; one that, while based principally on our preferred Strategy 1-2 combination, will permit adequate recourse to Strategies 3 and 4 when trying to prove difficult theorems.

As well, we believe a different strategy is called for; one that, while based principally on our preferred Strategy 1-2 combination, will permit adequate recourse to Strategies 3 and 4 when trying to prove difficult theorems.

4.1.2. Adopting the Right Attitude

There are several aspects to determining which part of $\text{psf}(\alpha)$ to construct and examine next. The most important aspect, given that Kripke implements a decision procedure and so can decide questions positively and negatively, is the attitude that Kripke ought adopt to $\alpha$ - i.e., should Kripke try to prove $\alpha$, or to refute $\alpha$. This difference in attitude is as important to automated theorem-proving as it is to question-answering by people. Adopting the right attitude makes a difference. For example, if $\alpha$ is unprovable, then Kripke would be well advised to try within the context of some proof-attempt to expand those nodes most likely to undermine that attempt; that is, not to search for closed subtrees, but to seek out the earliest structural weakness or the shortest unclosable AND-linked paths in the given proof-attempt. As well, Kripke should be cynical about the provability of most multisets it examines, and as a consequence, be both rigorous and vigorous about checking for all of the global properties of provable multisets. For example, by having a large set $S$ of matrices used in checking for the matrix property. As we have indicated earlier, Kripke should totally avoid Strategy 3 or 4 moves if $\alpha$ is
indeed unprovable - they will only complicate the picture and place inordinate space constraints on the process. On the other hand, if $\alpha$ is indeed provable, then KRIPKE would be best advised to adopt a vastly different approach. The question becomes not one of how to dispense with each and every proof-attempt in the fastest way, but of homing in on a (or the) proof. Searching for closed subtrees becomes the object of the exercise. Occasionally it may be desirable for KRIPKE to postpone further expansion of open paths in the current proof-attempt, and try a radically different approach. Strategy 3 or 4 moves might occasionally be desirable. As well, KRIPKE should be less cynical about the provability of most multisets it examines, and thus be less vigorous and rigorous in checking for the global properties; for example, by having $S$ small or even empty, or perhaps by occasionally omitting to check for any of the global properties and in preference making a quick thrust at a proof.

The attitude KRIPKE adopts to a given $\alpha$ should, of course, be amenable to interactive influence by the user. But with genuinely hard questions, such as the provability of formulas of Asset, KRIPKE must be able to adopt its own attitude, and be able to modify this attitude in the process of examining $pst(\alpha)$.

In order to make these assessments of attitude, we will need two sorts of information; one concerning the static status of a node (our costing information used in the previous subsection) and the other concerning the dynamic status of the node (i.e. assessments of the larger proof search tree of which the node is a part). We have discussed static assessments already. The dynamic assessment of a node depends on the context of the node. To illustrate, consider the segment of some $pst(\alpha)$ given in Figure 4-2. Static assessments of the nodes therein are given in parentheses, in the range 0 to 1, with low values being the good guys. On a simple static assessment, $\beta$ would be suggested as the next node to expand. As $\beta$ is AND-wise linked to $\beta$, our proving $\beta$ will not suffice to prove $\alpha$; $\beta$ must also be provable. On the other hand, a proof of $\gamma$ on its own will suffice to prove $\alpha$. If we take all this into account, it would seem that within this context $\gamma$ is the preferred node. Accounting for such situations is the task of dynamic assessments of nodes.
Let us imagine, now, a couple of scenarios, which will help us determine factors involved in KRIPKE's determining an attitude to $\alpha$. Assume firstly that KRIPKE has taken an optimistic attitude to the current proof-attempt for $\alpha$. Within this global attitude, if a node attracts a bad dynamic assessment, then it gets marked as obstinate and thus penalized by subjecting it more rigorously to various global property checks. This assessment affects the dynamic assessments of other nodes in this proof-attempt, especially those nodes in the immediate subtree on the node, and when the dynamic assessment of some fragment of $pst(\alpha)$ becomes intolerable, KRIPKE will retain the incompletely open subtree but will back up through $pst(\alpha)$ and select an alternative principal constituent (in effect, Strategy 4), or an alternative premiss-set (in effect, Strategy 3), for this or some node earlier in the proof-attempt, in an attempt to find a proof.

If KRIPKE finds itself having to examine nodes in an obstinate segment of $pst(\alpha)$ then it adopts a pessimistic attitude to the multisets in that segment and the strategy will alter. Thus if $\alpha$ itself, or an essential premiss for $\alpha$ such as one linked to $\alpha$ via a series of invertible rules, gets tagged as obstinate, then KRIPKE will take on a globally pessimistic attitude. It will never back up to find alternative principal constituents or alternative premiss-sets, will by and large expand the least.
(statically) promising premiss of a set of premisses in an effort to find a structural weakness in the proof-attempt (in effect, a subcase of Strategy 1), and will devote more resources to checking for various of the global properties.

So much for general scenarios. The catch comes in making the right static and dynamic assessments, picking the right nodes to class as obstinate, and making the right assessments as to when this obstinacy is intolerable enough to warrant a change of strategy, or more radically, a change of attitude. Just as important, KRIPKE must be able to change back from a pessimistic attitude to an optimistic one when the areas of obstinacy are removed, say, or become tolerable again. Suffice to say that we have not succeed in finding the right combination of factors. The machinery for handling changes of strategy or changes of attitude is not complicated; the trick is knowing when to make the change. We will return shortly to the matter of making assessments of attitude, pausing here to describe the modifications that can be made to Algorithm 2 to accommodate shifts in strategy.

To handle Strategies 3 and 4, we note that backing up through the tree to some node n in an open subtree, storing the subtree and making a new selection of premisses or even a new principal constituent, is not hard. The only (mildly) interesting technical question concerns remembering to examine this subtree should our alternatives themselves fail to produce a closed subtree on node n. To accommodate this within Strategy 2, we attach secondary states to nodes in \( \text{pst}(\alpha) \), with the secondary node-states being regular and paged. The primary node-state cannot become ‘failed’, as a result of failing to generate a closed subtree on node n, while the secondary state is ‘paged’; rather, if such an attempt to set the primary state is made, the stored subtree on node n is re-attached to n, n’s primary state is set to ‘open’, and if no other subtrees on node n are stored the secondary state is set to ‘regular’ and otherwise stays ‘paged’. Note that if node n is failed by some extended application of global property checks, or is part of an AND-linked subtree that is failed on another path as in the case of the right-side node of Figure 4-1, then we do not have to examine the stored subtrees on n; the source of failure is above the node, so to speak, and not related to any subtree on it.
Algorithm 3

program KRIPKE;

{Type and Var declarations as for Algorithm 2;
add secondary node_state fields (called secstate),
variables and fields for handling dynamic costings,
attitudes and so on}

function PROOF(var n : node_ptr) : node_state;
begin
{As for Algorithm 2, except when we come to replacing
failed subtrees, we have the following:}

if state=failed
then
if GET_NEXT_PREMISES(n) {tried ALL prem-sets? }
then state := open { no; reset state}
else case secstate of { yes; any stored? }
regular: FREE(n); { no; free space}
paged : begin { yes;
RESTORE(n); {get stored subtree;
update secstate}
state := open; {change state of node}
proof := open; {re-open path}
end;
end; {of cases}
proof := state; {return value for this call of PROOF}
end; {of proof}

{main program mainly as for Algorithm 2}

As the business of making dynamic assessments can be a substantial
costing, dynamic assessments of the current fragment of $pst(\alpha)$
can only realistically be updated from time to time. Fortunately, this fits in well
with our preferred control strategy of making recursive scans of $pst(\alpha)$. At the end
of a scan, attitudes can be formed regarding various proof-attempts in the current
fragment of $pst(\alpha)$. We omit the details.

We return now to our main problem of gauging when a dynamic assessment of a
node should be regarded as bad enough to treat the node as obstinate, and when
the obstinacy of a subtree should be classed as intolerable. Mistakes regarding the
former will slow the search down considerably, because nodes mistakenly classed as obstinate will attract considerable resources trying in vain to refute a provable node. Mistakes regarding the latter, involving looking at several proof-attempts in parallel for what is in fact an unprovable input multiset, are perhaps not so bad, given sufficient storage space. But to the extent that intolerance is determined in our scenarios by levels of obstinacy, a mistake concerning intolerance could well be premised on mistakes concerning obstinacy. If we could sever the dependency of assessments of intolerance from those of obstinacy, we could make changes of attitude independently of the number of obstinate nodes. Without having to contend with the time-penalities involved in dealing with a large number of obstinate nodes, attitude changes could become more feasible. In the closing parts of this section we mention one attempt to base changes of attitude on information other than that provided by static and dynamic assessments.

Analysis of the statistics provided in Appendix A seems in places to suggest that search spaces for theorems carry tell-tale features while those for non-theorems seem to have different characteristic properties. Some of these properties would, however, seem to be related to some intuitive assessment of the difficulty of the theorem or non-theorem (i.e. related to whether it belongs to say Standard or Asset.) The problems involved with forming attitudes, based on the degree of congruence between idealized and actual search spaces, lie not in the fact that statistical properties of observed proof search trees are being used normatively to typify and classify a current proof-attempt - this is after all a common and acceptable scientific method. Rather, the problems lie in assessing the intuitive ‘difficulty’ of a formula\(^1\), assessing the reliability of one’s idealized search spaces, and so on. Nonetheless, we believe that these ideas could be profitably explored. Either way, we take this opportunity to discuss some aspects of KRIPKE’s performance.

\(^1\)Size of formula, number of distinct propositional variables, and other common measures are inadequate in the case of LR, as the discussion in Chapters 2.2.5 and 3.2 indicates.
The observed differences between search spaces, comparing those for LR theorems to those for non-theorems, are:

- with theorems, the percentage of total nodes that are pruned due to failing some global property increases with difficulty, with the percentage rising from about 30% for the simple theorems of Standard to about 50% for the theorems of Asset. With non-theorems however, the direction seems to reverse, going from about 50% for simple non-theorems to 30% for the non-theorems of Asset.

- it is not surprising that the average time required to find a proof of a theorem increases with the difficulty, but what is surprising is that the time taken to refute more difficult non-theorems, like those of Impset, is on average significantly less than that taken to refute the average member of Standard.\(^1\)

- derived axioms always seem to feature in the search space of any theorem, although the proportion of total nodes that are axioms decreases slightly with increase in the difficulty of the theorem. With non-theorems though, the proportion of total nodes that are axioms, derived or otherwise, is always insignificant.

- the applicability of the Ko rule for effecting idempotence reduction, while rare for simple theorems, is reasonably common for more difficult theorems. In the case of non-theorems, idempotence reduction is universally rare.

We are reluctant to conclude much from these observations, apart from reiterating the view that certain classes of formulas would seem to give rise to search spaces with characteristic properties, and that monitoring a proof-attempt for these properties could assist in forming the right attitude.

\(^1\)This remains the case even if we exclude the non-theorems of each set which took the greatest individual times, thereby reducing average times to 2927/64 = 50 msec. for Standard and 3521/106 = 34 msecs. for Impset.
4.2. Bits and Pieces

In this final section of the thesis we provide the more central details concerning KRIPKE’s data structures, discuss some aspects of rule applications, assess the utility of KRIPKE’s node-pruning techniques, and briefly discuss the matter of program correctness.

4.2.1. Data Structures - or, Roman Numerals and Gödel Numbers are fine, but...

While often, in the implementation of any algorithm, there will feature a myriad of distinct types of data, we will confine our discussion here to those of central concern in KRIPKE; i.e. binary trees of nodes which contain multisets of formulas. The techniques of representing binary trees can be extracted from any text on data structures, and we simply note concerning them that, apart from containing a multiset, the nodes of a tree contain pointers to other parts of the tree, costing information, the node-state, and various control variables. Of especial interest is the way KRIPKE represents multisets of formulas.

Recalling that L5 has the subformula property, defined by Definition 2.10, we can anticipate the formulas that will occur as members of multisets in \( \text{pst}(A) \) by producing the parse-tree for \( A \), as formula \( A \) is input. We may then form a table of subformulas, from this parse-tree, and from then on refer to a particular subformula by way of its location in this table. We will call this table the pre-analysis table. Various properties of some subformula, \( B \), may be determined by examining the parse-tree. For example, \( B \)'s immediate subformulas (again, as integer pointers into the pre-analysis table), the types of connectives that occur in \( B \), the positive and negative parts of \( B \), various normal-formed equivalents of \( B \), and so on, can be determined by reference to the parse-tree and stored along with \( B \) in the pre-analysis table. Doing so will considerably ease the task of determining various properties of multisets; for example, most of the global properties we desire of multisets in \( \text{pst}(\alpha) \) are determined by whether some or all of the members of the multiset have the property.
Another advantage in constructing a pre-analysis table is that, as most simplified formulas will have few subformulas, we will often be able to represent the integer pointer to some subformula within a byte of few bits, and pack several such pointers within the one word of memory. We will always sort the entries in the pre-analysis table, with the primary key being the main connective of the subformula (in the order +, &, V, o followed by literals, so that fission and conjunction formulas, or invertible formulas, occur before the others). Within this primary ordering, entries in the pre-analysis table may be further sorted in terms of some function of the degree and outer degree of the subformulas, defined in Definitions 2.23 and 2.28 respectively. Thus, if the members of a multiset, being some collection of integer pointers into the pre-analysis table, are ordered numerically, and candidate principal constituents are selected in turn as they occur in this ordering, then not only will invertible rules always be applied before non-invertible rules, but within a given rule-type, say $Po^n$, the rule will be applied to less complex fusion formulas before more complex. This latter process, of expanding nodes in terms of less complex principal constituents before more complex ones, is a simple but effective way of 'happenning' quickly upon violations of, for example, the strict positive/negative parts property. Having thus fixed our representation of formulas in terms of integer pointers into a pre-analysis table, we turn now to the matter of representing multisets of formulas.

Naturally, the way one represents a given datatype depends on the kinds of operations one wishes to perform on the data. Obviously, different representations suit different purposes, and occasionally one has apparently incompatible operational requirements of the one kind of data. At least, in the sense that one manner of representation might be optimal for one operation, while a vastly different representation might be optimal for another; alternatively, one might be optimal with respect to time considerations, while another is optimal space-wise. The datatype, multiset, faces exactly these difficulties, at least given the operational requirements we need to place on multisets within the context of KRIPKE. We present several alternative representations of multisets, and in each case illustrate it's representation of the multiset $[A,A,B,B,B,C]$:
(a) as a sequence of formulas, ordered in accordance with the conventions of the previous paragraph, with repetitions of a generator occurring adjacent to each other - e.g. \(<A,A,B,B,B,C>\); 
(b) as a pair of sequences, the first being an ordered list of the generators, and the second being the list of corresponding counts of the number of times each generator occurs - e.g. as \(<A,B,C>\) and \(<2,3,1>\); 
(c) given that the range of generators is determinable at the time of constructing the pre-analysis table, then we may represent a multiset in this context as a list of the number of times it occurs in the given multiset \(\beta\) - e.g. assume that there are 10 distinct generators possible, with say A being the first, B the third and C the ninth, then our multiset would look like \(<2,0,3,0,0,0,0,0,1,0>\).

There are other variations which mix components of (a), (b) or (c), or which specify that a sequence be a packed array, or a linked list, and so on, but (a)...(c) seem to be the basic alternatives. To be precise, (a) captures the basic notational schema for multisets implicit in Definition 2.1, (b) reflects our occasional recourse to using exponential notation, and (c) seems to capture Meyer's idea in [Meyer 73] of treating multisets as products of primes (with a bit of positional notation thrown in).

Now, the operations we most often wish to perform on multisets would seem to fall into three broad categories:

(A) listing operations, such as displaying the members of a multiset, selecting the next member so as to collect, say, the positive parts of all the members, and so on;
(B) multiset comparison operations, such as multiset identity, and the multiset containment operations \(|\) and \(;/\);
(C) modifying a multiset's contents, typically involved in applying some rule to a copy of a multiset, and thus including various insertion, deletion and masking operations.
By and large, it appears that (A)-operations go best with (a)-style representations, (B) with (b), and (C) with (c), although there is little to distinguish (a) and (b) regarding the (A)-operations, for mostly we are only interested in selecting or collecting next generators, rather than next members.

Representation (b) is the staple representation used by KRIPKE, although a given multiset will be represented at various stages in its life in \( pst(\alpha) \) by either or both of the other two styles also. Apart from the fact that the (b)-style is the most versatile, its memory overheads are not significantly greater than those of the (a)-style, and significantly less than the (c)-style. Moreover, once a multiset has been placed in \( pst(\alpha) \), the most frequent operations performed on it are those that involve comparing successor multisets against it for the Curry property, or examining it for containment by newly derived axioms and the like, and this is exactly (b)'s strength.

4.2.2. Applying Rules

One of the principal virtues of a Gentzen formulation of a logic is that each rule more or less states an appropriate algorithm for actually applying the rule to a consecution. In the context of proof search trees these rule applications are reversed, as we have noted elsewhere, in an attempt to find premisses for conclusions rather than to produce conclusions from previously derived premisses. But again, the algorithms would seem obvious.

Closure rules, for recognizing multisets that are axioms, terminate construction of a path in the \( pst(\alpha) \), because an axiom by definition has no premisses. In the notes to Algorithm 1, we have discussed the rules \( Ko \) and \( Kf \). Turning to the connective rules, we see that conclusions of the other invertible rules, \( P+ \) and \( P\& \), determine a unique premiss-set once the principal constituent for the application of \( P+ \) or \( P\& \) has been fixed. In the case of \( P+ \), its unique premiss-set has just one premiss. In the case of \( P\& \), its premiss-set has two AND-linked premisses. Determining premiss-sets for \( \alpha = [AvB, \beta] \), with (AvB) principal and where \( \alpha \) was
derived using the L5-rule $Pv''$, is easy: if $\beta$ contains a copy of $(AvB)$, then there are exactly two candidate premiss-sets for $\alpha$:

$$[\beta,A] \quad \text{and} \quad [\beta,B]$$

and if $\beta$ does not contain a copy of the principle constituent $(AvB)$, then we may add another two possible singleton premiss-sets:

$$[\beta,A,AvB] \quad \text{and} \quad [\beta,B,AvB]$$

The matter of fixing a set of premiss-sets for $\alpha$, where we assume that generator $A$, of the form $BoC$, is principal in $\alpha$ under the rule $Pd''$, is not so easy. However, our discussion of the complexity of the $Pd''$ rule, in Chapter 2.2.5, provided a constructive algorithm for counting premiss-pairs for the $Pd''$ rule, so we can construct our set of premiss-sets using that algorithm.

We remind the reader that because our preferred strategy, using Algorithm 2, only involves having one principal constituent for the multiset of each open node active at any one time and moreover only permits having one premiss-set, based on that principal constituent, active at any one time, rules that have more than one candidate premiss-set, such as $Po''$ or $Pv''$, will have to be applied in a step-wise fashion. Again, we omit details.

4.2.3. Checking for Local and Global Properties

Checking for the Curry property involves checking for the appropriate multiset containment relations between some node $n$ of $pst(\alpha)$ and every node in its upward path to $\alpha$. Thus every time a node is created, we have to traverse the upward path. Despite intuitions, this is a relatively inexpensive check in the total context of KRIPKE, though in isolation it would be properly considered a potentially time-wise crucial problem. Checks to see whether each newly created node is a member of the list of axioms, or of the list of obstinate nodes, could be viewed similarly. Although attention to the usual ways of optimizing list look-ups will repay the effort, at least in terms of mildly improved runtimes, such efforts do not lead to deciding otherwise undecided formulas. We mention, however, two optimizations which are of special interest in the context of ATP.
The first concerns checks for the Curry property. This property features in KRIPKE solely to guarantee decidability; in particular, to prevent subtle and useless repetitions of multisets in any given path. The property is expressed in terms of all nodes in $pst(\alpha)$, and so it seems desirable to check for the property as each node is created. Yet, no harm occurs, and decidability is not jeopardized, if $pst(\alpha)$ is examined for the property not as each node is created but after longer, regular intervals. To effect this, we mark each node in our fragment of $pst(\alpha)$ to indicate whether it has been checked with respect to its upward path for the property. On every $i$th scan, we check the next $j$ unmarked nodes in every path, and mark them, where $i$ and $j$ are constants. Of course, the shape of our fragment of $pst(\alpha)$ will alter from one scan to the next, but these alterations will always be in the lowermost subtrees of the fragment, and so will not affect matters; although some marked nodes may be deleted or replaced, no node added since the last check will have been marked. If new nodes frequently led to $pst(\alpha)$ failing the Curry property, then this alternative method of effecting the check would be no improvement over, and perhaps worse than, the initial simple method of checking $pst(\alpha)$ for the property. As it happens, the statistics of Appendix A reveal that failure to have the Curry property is rare, and so our alternative method will often lead to some time-wise improvement in KRIPKE's performance.

The second optimization relates to checking for axioms or obstinate nodes. Again, the obvious way of doing this is to check the respective lists as each node is created and added to $pst(\alpha)$, and again the alternative, interval-method of of the last paragraph is available here. Here though we prefer the simple method over the interval-method; for one thing, we want to close paths as soon as possible. Given, then, that we will want to check each new node against each list, it will help if the lists are as small as possible.\(^1\) We could of course set arbitrary limits to the number

\(^1\)We have in mind here a simple item-by-item check against members of the list. There are other methods, such as hashing, which involve storing and retrieving items using locations which are (partial) functions of the item itself. Unfortunately, the obvious hash-keys for multisets, such as the underlying generating set, do not give appropriately distributed ranges for multisets in the context of KRIPKE, and so the so-called collision problem is especially bad. On balance, hashing does not seem to repay the effort, although we admit to not having explored the possibilities extensively.
of derived axioms or obstinate nodes to remember, and sometimes we will have no alternative but to do so. However, by paying attention to logical features of multisets, we can eliminate essentially redundant information. Because \( W \) is a legitimate principle of LR, we know that if a multiset 

\[ [A^p_1, ..., A^p_k] \]

is L5-provable, then so too will every multiset of the form 

\[ [A^q_j, ..., A^q_k] \]

where \( 1 \leq j_i \leq n_i \).

Indeed, this principle is the logical support for the Curry property. As a consequence, our list of axioms need only contain multisets which are \textit{maximal} on a set of generators, where a multiset is so maximal iff no other multiset in the list strongly contains it. When we come to check for axiomhood against members of our list, it will suffice then for \( \beta \) to be provable if \( \beta \) is identical to \textit{or strongly contained in} some member of the list. Conversely, if 

\[ [A^p_1, ..., A^p_k] \]

is \textit{not} L5-provable, then so too will every multiset of the form 

\[ [A^q_j, ..., A^q_k] \]

where \( 1 \geq j_i \geq n_i \).

be L5-unprovable. When it comes to adding members to our black-list of obstinate nodes suspected of being unprovable, we need only keep multisets which are \textit{minimal} (in the obvious sense) on a set of generators. When we come to check whether some node should be classed as obstinate and penalized because it has been added to our black-list at some time, it will suffice to so penalize it if it is identical to \textit{or strongly contains} some member of the list.

Turning to global property checks, we note that by and large the checks are simple and so we omit algorithmic descriptions of them.\(^1\) Nonetheless, we will take the opportunity of assessing the utility of making these checks as part of the general strategy. We have recently noted that the Curry property checks are responsible for rejecting few nodes from search spaces. In the absence of global properties, however, the task of pruning paths in \( pst(\alpha) \) would fall totally on checks

\(^1\)Chapter 3 discussed the matrix property at some length.
for the Curry property. We take the fact that the Curry checks are so rarely appealed to, then, as a forceful indication of the general utility of checking for global properties, and the individual utility of the global properties we have considered here. Supporting this claim, we note again that checks for global properties are responsible for pruning often up to half of all nodes created.\(^1\)

The checks for each (active) global property occur in a fixed order, with this ordering being based on how much time the check takes. After all, a multiset need fail only one such check to warrant pruning, so the cheaper the check the better. The order of checks is:

- strong and weak positive/negative parts
- rule-of-2
- strict positive/negative parts
- matrix property

with the first three checks taking about the same time each. Assuming this order, the following statistical observations can be extracted from Appendix A. With simple theorems, the earlier checks - especially the strong/weak parts checks - do nearly all of the node-pruning work, matrices do a little, and strict parts tests and rule-of-2 tests do nearly nothing. In comparison, with the most difficult theorems, the later checks - matrices and strict parts - are responsible for nearly all of the node-pruning work. With non-theorems, nearly all of the work is nearly always done by matrix checks.

In conclusion, then, the matrix property is universally useful, although time-wise often expensive. Perhaps the best value for money, though, comes from the strict parts property.

---

\(^1\)See Appendix A, and the previous section.
4.2.4. Program Correctness

Most of the burden of showing KRIPKE correct is, for all intents and purposes, already accommodated by our arguments for the adequacy of the decision procedure provided in Chapter 2. Thus we have a guarantee that our algorithms, in the large, are correct. This is not of course equivalent to claiming either that KRIPKE is free of infinite loops or *will* always, at least given large enough space and time, terminate for all inputs. Nor is it equivalent to the claim that even should KRIPKE always terminate, it would always terminate with the correct answer. Both of these conditions would be unintended errors, and would point to a faulty implementation of a logically adequate algorithm.

The question of whether a piece of code is a correct implementation of some algorithm - whether a program meets its specifications - will sometimes be amenable to answer via the techniques of program verification. The techniques vary, although the common core would invariably seem to involve stating the basic algorithm in terms of some first-order theory that can model dynamic constructs - such as assignment to variables, recursion and other control mechanisms such as while-loops - and data structures, and then prove that certain other statements in the theory (say, that the program terminates with certain variables having certain values) follow in the theory from the statement of the basic algorithm. Even for rather elementary programs, however, the verification-language statements modelling the program can be quite lengthy and intricate, and as a consequence, many researchers in the area have concentrated on methods for automating the construction of such statements, and then using ATP systems for proving appropriate properties of these. As the discussion in [VERkshop 80] demonstrates, many of the dynamic constructs that feature in a program like KRIPKE would still seem beyond the grasp of current automated program verifiers, and so these techniques are unavailable to us.

Although we have thus been unable to guarantee that KRIPKE is free of the dreaded Dijkstra Deep Bug, the question of the fidelity of KRIPKE to our decision procedure for L5 was of sufficient concern to lead us to taking the following steps.
Of primary concern was the matter of soundness - if KRIPKE claims \( \tau \) as a proof of \( \alpha \), then \( \tau \) is in fact such a proof. To reassure us of soundness, every proof output by KRIPKE is subjected to examination by an independent automated proof-checker. In order to beg as few questions as possible, the checker ignores all information about how the proof was actually found within the context of \( \text{pst}(\alpha) \), such as which members of the multisets in \( \tau \) were principal, and treats \( \tau \) simply as a tree of consecutions. The proof, \( \tau \), is then traversed, from tips to root - i.e. from putative axioms to the input multiset - checking that the tips are indeed \( \text{L5} \)-axioms, and that every other multiset in \( \tau \) indeed follows, using some rule, from the given premisses.

The related question of completeness - if KRIPKE claims that \( \alpha \) is not \( \text{LR} \)-provable, then in fact there is no \( \text{L5} \) proof of \( \alpha \) - is considerably more difficult to check for, because it amounts to the claim that not only has each potential proof constructed by KRIPKE failed to be a proof, but that KRIPKE has investigated all possible ways of trying to prove \( \alpha \). Various checks and balances have been built into the program, so that if the Djikstra Bug does lie on the side of completeness, we will be made aware of it. One set of checks are invoked during the step-wise development of any version of KRIPKE, or during the modification of any resident version, and involve testing a large set of formulas whose status as \( \text{LR} \) theorems or non-theorems is already known, either from the literature or from previous runs of the program, to ensure that the new version concurs. Another set of checks reside permanently in KRIPKE. Some of these involve detecting when \( \text{pst}(\alpha) \) enters some logically impossible state; e.g. applying the \( \text{Po}^" \) rule to a multiset that contains a fission formula. Others involve matching some micro-level behaviour of KRIPKE with the expected behaviour; e.g., whether the number of times the \( \text{Po}^" \) rule has in fact been applied to some multiset \([\text{AoB}, \alpha]\), say, with AoB principal, agrees with the value given by our premiss-set counting function \( \text{Cnt} \), from Chapter 2.2.5. In order to make such checks as unbiased as possible, alternative (albeit formally equivalent) functions, are often used to effect the check. For example, an operation \( \text{op}(\alpha) \) on multisets may be effected one way, by taking \( \alpha \) represented data-wise as a
sequence of formulas - e.g. \([C,A,B,A,C]\) - and alternatively by taking \(\alpha\) represented as a set of indexed generators - e.g. \(\{A^2,B^1,C^2\}\).

Our final concern regarding program correctness, given that we are implementing a decision procedure, must be that KRIPKE will, at least in principle, always terminate. This question is not determined by having joint guarantees of soundness and completeness, for although KRIPKE may look at all and only potential proofs of \(\alpha\), and may apply all rules adequately, some program control bug might, in bizarre conditions related partly to the content of \(\alpha\), enjoin KRIPKE to loop infinitely over some part of \(pst(\alpha)\). To some extent, this is of least concern, for there will always be a range of inputs for which KRIPKE will not terminate within practical time and space limitations, and thus for all practical purposes, KRIPKE will have a three-valued metalogic:

- \(\alpha\) is provable
- \(\alpha\) is not provable,
- and "can you afford to wait a little bit longer".

We could, for instance, remember the entire state of \(pst(\alpha)\), and from time to time check that the current state of \(pst(\alpha)\) was not identical to some remembered state. These sorts of checks do not amount to a decision procedure for detecting circularity, and moreover, the Law of Diminishing Returns would seem to militate against making this sort of check anyway.
Appendix A

Runtime Statistics for KRIPKE

The following tables contain information regarding three different runs of KRIPKE against three different sets of formulas:

**Standard**

The first set of formulas is the standard test set mentioned in Chapter 4.2. It contains all the axioms of \( LR \) given in Table 2-1, and a range of formulas extracted from the literature. Included, for example, are

\[
((p \rightarrow (q \rightarrow r)) \& (q \rightarrow (p \vee r))) \rightarrow (q \rightarrow r)
\]

\[
q \rightarrow r \rightarrow (p \rightarrow (q \rightarrow r \rightarrow s) \rightarrow (p \rightarrow s))
\]

We note that KRIPKE can decide the theoremhood or otherwise of all of the 169 formulas in **Standard**.

**Impset**

The second set contains all possible implications between the sixteen formulas in Table 1-1 - i.e. 256 implicational formulas. For example, one such formula, namely the implication between C1 and C2 is

\[
po(p+p)0(qv \sim q) \rightarrow po(qv \sim qv(\sim po \sim p))
\]

If C1 implies C2, and conversely, then C1 and C2 are of course provably equivalent. We can then partition the set of defining formulas of Table 1-1 into sets of provably equivalent formulas, as we indicated at the end of Chapter 1.3. In the following structure each point consists of the set of candidate defining formulas which are provably equivalent in \( LR \), and the order relation on the structure is OR-provable implication:
Kripke can decide the theoremhood or non-theoremhood of each of the 256 formulas in $\text{Impset}$.

$\text{Asset}$ and the third set contains the statements of associativity mentioned in Table 1-1. Rather than treat this set as 16 equivalences, we have divided each equivalence into its pair of implicational formulas. Although each of these 32 formulas is in fact a theorem of $\text{LR}$, Kripke does not terminate with a proof in all cases within reasonable time and space constraints. Formulas for which Kripke fails to terminate are classed as $\text{incomplete}$.

These sets represent formulas of increasing complexity, and cover the range of formulas that Kripke can decide. We further divide each set of formulas into several subsets - theorems, non-theorems, and incomplete - and we provide statistics not only for each set of formulas, but for each applicable subset. For each such subset, we provide two different tables of runtime statistics.

For the purposes of standardizing these statistics, the version of Kripke based on Algorithm 2 from Chapter 4 was used, checks for all of the global properties were active for the whole duration of each run, and the set of matrices used for the matrix property was the singleton set containing the $\text{CHAIN5}$ matrix detailed in Figure 3-1.

Table A gives three types of statistic for various fields. The three types are: the average number per formula (or theorem, or non-theorem, or incomplete formula, as appropriate), the maximum in any one formula, and total for all formulas (or theorems, etc.). The fields are:

- No. of recursive passes: the number of recursive sweeps of $\text{pst}(\alpha)$ that are made, described in Chapter 4.1.
<table>
<thead>
<tr>
<th>Time (in msecs.)</th>
<th>the CPU time taken in 1000th’s of a second</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of nodes created</td>
<td>the number of nodes that take on the status <em>linked</em> as described in Chapter 4.1.</td>
</tr>
<tr>
<td>linked</td>
<td>the number of obstinate nodes.</td>
</tr>
<tr>
<td>penalized</td>
<td>the number of nodes that are created and recognized as simple axioms of the form $Ax_p$, $Ax_t$ or $Ax_T$.</td>
</tr>
<tr>
<td>as axioms</td>
<td>the number of nodes that are recognized as derived axioms in the sense of Chapter 2.3.2.</td>
</tr>
<tr>
<td>derived axs</td>
<td>the number of nodes that are pruned from $pst(\alpha)$ because they fail to have one of the global properties of provable multisets detailed in Chapters 2.3 and 3.</td>
</tr>
<tr>
<td>pruned</td>
<td>the number of times that the first-selected principal constituent for some multiset in $pst(\alpha)$ has failed to produce a closed subtree (for reasons other than failing local or global properties - i.e. by total exhaustion of possible rule applications) and an alternative principal constituent has been selected.</td>
</tr>
<tr>
<td>No. of alternative pc.</td>
<td>the number of times that some multiset has taken on the status <em>failed</em> for reasons other than failing some local or global property - i.e. by having exhausted all possible rule applications for all possible principal constituents.</td>
</tr>
<tr>
<td>Failed: no altern. pc.</td>
<td>the number of distinct multisets that are being remembered as possibly obstinate multisets to be penalized when and if they are detected.</td>
</tr>
<tr>
<td>Size of Black list</td>
<td>the number of distinct multisets that are being remembered as derived axioms.</td>
</tr>
<tr>
<td>White list</td>
<td></td>
</tr>
</tbody>
</table>

The table provides two general types of statistics. It gives the number of formulas where the percentage of nodes in the second search run time for the given time of statistic gives the percentage of all nodes in all proof search runs for all formula having the property of provability. The left-hand fields are the CPU time taken in 1000th’s of a second and have obvious interpretations.
Final Proof: nodes these fields only apply to the subset of theorems and have obvious interpretations.

width

depth

Table B provides two general types of statistic. The first type gives the number of formulas where the percentage of nodes, in the proof search tree for the given formula, having the property described in the field to the left reaches 1%, 10%, 20% or 50% of all nodes in the proof search tree for that formula. The second type of statistic gives the percentage of all nodes in all proof search trees for all formulas in the subset having the property described in the field to the left. The properties in the left-hand fields are:

- Failing Curry property
- Failing Strong/Weak parts
- rule-of-2
- strict parts
- Matrices
- Total percentages for failures against any global property
- Idempotent Reducts percentages for multisets to which the rule $Ko$ is applied to produce a simpler multiset
- Axioms: total percentages of nodes that are recognized as either simple or derived axioms
- derived percentages for nodes recognized as derived axioms
- Failing no altern. pc. percentages for failures other than by failing to have some local or global property
- Alternative pc. percentages of nodes where a new principal constituent was selected.
TABLE A, for the 104 Theorems of Standard

<table>
<thead>
<tr>
<th></th>
<th>Average/form</th>
<th>Maximum</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of recursive passes</td>
<td>2</td>
<td>5</td>
<td>198</td>
</tr>
<tr>
<td>Time (in msecs.)</td>
<td>244</td>
<td>4701</td>
<td>25412</td>
</tr>
<tr>
<td>No. of nodes created</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>linked</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>penalized</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>as axioms</td>
<td>1</td>
<td>3</td>
<td>61</td>
</tr>
<tr>
<td>derived axs.</td>
<td>1</td>
<td>4</td>
<td>143</td>
</tr>
<tr>
<td>pruned</td>
<td>3</td>
<td>40</td>
<td>350</td>
</tr>
<tr>
<td>No. of alternative pc.</td>
<td>0</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Failed: no altern. pc.</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Size of Black list</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>White list</td>
<td>5</td>
<td>12</td>
<td>530</td>
</tr>
<tr>
<td>Final Proof: nodes</td>
<td>5</td>
<td>12</td>
<td>543</td>
</tr>
<tr>
<td>width</td>
<td>2</td>
<td>4</td>
<td>203</td>
</tr>
<tr>
<td>depth</td>
<td>4</td>
<td>9</td>
<td>418</td>
</tr>
</tbody>
</table>

TABLE B, for the 104 Theorems of Standard

<table>
<thead>
<tr>
<th></th>
<th>No. of formulas where % reaches</th>
<th>% of all nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+1%</td>
<td>+10%</td>
</tr>
<tr>
<td>Failing Curry property</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Failing Strong/Weak parts</td>
<td>48</td>
<td>46</td>
</tr>
<tr>
<td>rule-of-2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>strict parts</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>matrices</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>48</td>
<td>46</td>
</tr>
<tr>
<td>Idempotent reducts</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Axioms: total</td>
<td>104</td>
<td>99</td>
</tr>
<tr>
<td>derived</td>
<td>104</td>
<td>80</td>
</tr>
<tr>
<td>Failing no altern. pc.</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alternative pc.</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>
**TABLE A, for the 65 Non-Theorems of Standard**

<table>
<thead>
<tr>
<th></th>
<th>Average/form</th>
<th>Maximum</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of recursive passes</td>
<td>2</td>
<td>15</td>
<td>103</td>
</tr>
<tr>
<td>Time (in msecs.)</td>
<td>200</td>
<td>10098</td>
<td>13025</td>
</tr>
<tr>
<td>No. of nodes created</td>
<td>11</td>
<td>374</td>
<td>744</td>
</tr>
<tr>
<td>linked</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>penalized</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>as axioms</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>derived axs.</td>
<td>0</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>pruned</td>
<td>6</td>
<td>272</td>
<td>397</td>
</tr>
<tr>
<td>No. of alternative pc.</td>
<td>0</td>
<td>8</td>
<td>17</td>
</tr>
<tr>
<td>Failed: no altern. pc.</td>
<td>0</td>
<td>4</td>
<td>19</td>
</tr>
<tr>
<td>Size of Black list</td>
<td>0</td>
<td>4</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>White list</td>
<td>1</td>
<td>6</td>
<td>85</td>
</tr>
</tbody>
</table>

**TABLE B, for the 65 Non-Theorems of Standard**

<table>
<thead>
<tr>
<th></th>
<th>% of all nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of formulas where % reaches</td>
<td>+1%</td>
</tr>
<tr>
<td>Failing Curry property</td>
<td>1</td>
</tr>
<tr>
<td>Failing Strong/Weak parts</td>
<td>15</td>
</tr>
<tr>
<td>rule-of-2</td>
<td>13</td>
</tr>
<tr>
<td>strict parts</td>
<td>4</td>
</tr>
<tr>
<td>matrices</td>
<td>34</td>
</tr>
<tr>
<td>Total</td>
<td>51</td>
</tr>
<tr>
<td>Idempotent reducts</td>
<td>0</td>
</tr>
<tr>
<td>Axioms: total</td>
<td>4</td>
</tr>
<tr>
<td>derived</td>
<td>4</td>
</tr>
<tr>
<td>Failing no altern. pc.</td>
<td>15</td>
</tr>
<tr>
<td>Alternative pc.</td>
<td>10</td>
</tr>
</tbody>
</table>
### TABLE A, for the 169 Formulas of Standard

<table>
<thead>
<tr>
<th>No. of recursive passes</th>
<th>Average/Max</th>
<th>Maximum</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>15</td>
<td>301</td>
<td></td>
</tr>
<tr>
<td>227</td>
<td>10098</td>
<td>38437</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>374</td>
<td>1792</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>66</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>152</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>272</td>
<td>747</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>615</td>
<td></td>
</tr>
</tbody>
</table>

### TABLE B, for the 169 Formulas of Standard

<table>
<thead>
<tr>
<th>No. of formulas where % reaches</th>
<th>% of all nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1%</td>
<td>+10%</td>
</tr>
<tr>
<td>Failing Curry property</td>
<td>2</td>
</tr>
<tr>
<td>Failing Strong/Weak parts</td>
<td>63</td>
</tr>
<tr>
<td>rule-of-2</td>
<td>13</td>
</tr>
<tr>
<td>strict parts</td>
<td>4</td>
</tr>
<tr>
<td>matrices</td>
<td>37</td>
</tr>
<tr>
<td>Total</td>
<td>99</td>
</tr>
<tr>
<td>Idempotent reducts</td>
<td>0</td>
</tr>
<tr>
<td>Axioms: total</td>
<td>108</td>
</tr>
<tr>
<td>derived</td>
<td>108</td>
</tr>
<tr>
<td>Failing no altern. pc.</td>
<td>15</td>
</tr>
<tr>
<td>Alternative pc.</td>
<td>14</td>
</tr>
</tbody>
</table>
TABLE A, for the 149 Theorems of Impset

<table>
<thead>
<tr>
<th></th>
<th>Average/form</th>
<th>Maximum</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of recursive passes</td>
<td>6</td>
<td>112</td>
<td>947</td>
</tr>
<tr>
<td>Time (in msecs )</td>
<td>1206</td>
<td>41846</td>
<td>179634</td>
</tr>
<tr>
<td>No. of nodes created</td>
<td>78</td>
<td>2753</td>
<td>11659</td>
</tr>
<tr>
<td>linked</td>
<td>1</td>
<td>17</td>
<td>192</td>
</tr>
<tr>
<td>penalized</td>
<td>0</td>
<td>73</td>
<td>73</td>
</tr>
<tr>
<td>as axioms</td>
<td>6</td>
<td>39</td>
<td>904</td>
</tr>
<tr>
<td>derived axs.</td>
<td>5</td>
<td>75</td>
<td>806</td>
</tr>
<tr>
<td>pruned</td>
<td>23</td>
<td>1632</td>
<td>3461</td>
</tr>
<tr>
<td>No. of alternative pc.</td>
<td>0</td>
<td>20</td>
<td>41</td>
</tr>
<tr>
<td>Failed: no altern. pc.</td>
<td>0</td>
<td>12</td>
<td>19</td>
</tr>
<tr>
<td>Size of Black list</td>
<td>0</td>
<td>12</td>
<td>19</td>
</tr>
<tr>
<td>White list</td>
<td>18</td>
<td>30</td>
<td>2649</td>
</tr>
<tr>
<td>Final Proof: nodes</td>
<td>22</td>
<td>45</td>
<td>3313</td>
</tr>
<tr>
<td>width</td>
<td>8</td>
<td>18</td>
<td>1266</td>
</tr>
<tr>
<td>depth</td>
<td>10</td>
<td>18</td>
<td>1451</td>
</tr>
</tbody>
</table>

TABLE B, for the 149 Theorems of Impset

<table>
<thead>
<tr>
<th></th>
<th>+1%</th>
<th>+10%</th>
<th>+20%</th>
<th>+50%</th>
<th>nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failing Curry property</td>
<td>20</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Failing Strong/Weak parts</td>
<td>24</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>rule-of-2</td>
<td>50</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>strict parts</td>
<td>89</td>
<td>16</td>
<td>6</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>matrices</td>
<td>122</td>
<td>51</td>
<td>10</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td>Total</td>
<td>136</td>
<td>79</td>
<td>29</td>
<td>7</td>
<td>30</td>
</tr>
<tr>
<td>Idempotent reducts</td>
<td>32</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>Axioms: total</td>
<td>149</td>
<td>145</td>
<td>81</td>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td>derived</td>
<td>149</td>
<td>50</td>
<td>30</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Failing no altern. pc.</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alternative pc.</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
**TABLE A, for the 107 Non-Theorems of \textit{Impset}**

<table>
<thead>
<tr>
<th>Category</th>
<th>Average/form</th>
<th>Maximum</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of recursive passes</td>
<td>1</td>
<td>12</td>
<td>129</td>
</tr>
<tr>
<td>Time (in msecs.)</td>
<td>44</td>
<td>1175</td>
<td>4696</td>
</tr>
<tr>
<td>No. of nodes created</td>
<td>9</td>
<td>166</td>
<td>933</td>
</tr>
<tr>
<td>linked</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>penalized</td>
<td>0</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>as axioms</td>
<td>0</td>
<td>7</td>
<td>14</td>
</tr>
<tr>
<td>derived axs.</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>pruned</td>
<td>2</td>
<td>73</td>
<td>251</td>
</tr>
<tr>
<td>No. of alternative pc.</td>
<td>0</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>Failed: no altern. pc.</td>
<td>0</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>Size of Black list</td>
<td>0</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>White list</td>
<td>3</td>
<td>5</td>
<td>302</td>
</tr>
</tbody>
</table>

**TABLE B, for the 107 Non-Theorems of \textit{Impset}**

<table>
<thead>
<tr>
<th>Category</th>
<th>No. of formulas where % reaches</th>
<th>% of all nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+1%</td>
<td>+10%</td>
</tr>
<tr>
<td>Failing Curry property</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Failing Strong/Weak parts</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>rule-of-2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>strict parts</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>matrices</td>
<td>105</td>
<td>105</td>
</tr>
<tr>
<td>Total</td>
<td>107</td>
<td>107</td>
</tr>
<tr>
<td>Idempotent reducts</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Axioms: total</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>derived</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Failing no altern. pc.</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Alternative pc.</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
TABLE A, for the 256 Formulas of *Impset*

<table>
<thead>
<tr>
<th></th>
<th>Average/form</th>
<th>Maximum</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of recursive passes</td>
<td>4</td>
<td>112</td>
<td>1076</td>
</tr>
<tr>
<td>Time (in msecs)</td>
<td>720</td>
<td>41846</td>
<td>184330</td>
</tr>
<tr>
<td>No. of nodes created</td>
<td>49</td>
<td>2753</td>
<td>12592</td>
</tr>
<tr>
<td>linked</td>
<td>1</td>
<td>17</td>
<td>192</td>
</tr>
<tr>
<td>penalized</td>
<td>0</td>
<td>73</td>
<td>81</td>
</tr>
<tr>
<td>as axioms</td>
<td>4</td>
<td>39</td>
<td>918</td>
</tr>
<tr>
<td>derived axs.</td>
<td>3</td>
<td>75</td>
<td>809</td>
</tr>
<tr>
<td>pruned</td>
<td>15</td>
<td>1632</td>
<td>3712</td>
</tr>
<tr>
<td>No. of alternative pc.</td>
<td>0</td>
<td>20</td>
<td>51</td>
</tr>
<tr>
<td>Failed: no altern. pc.</td>
<td>0</td>
<td>12</td>
<td>27</td>
</tr>
<tr>
<td>Size of Black list</td>
<td>0</td>
<td>12</td>
<td>27</td>
</tr>
<tr>
<td>White list</td>
<td>12</td>
<td>30</td>
<td>2951</td>
</tr>
</tbody>
</table>

TABLE B, for the 256 Formulas of *Impset*

<table>
<thead>
<tr>
<th>No. of formulas where % reaches</th>
<th>% of all</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+1%</td>
</tr>
<tr>
<td>Failing Curry property</td>
<td>22</td>
</tr>
<tr>
<td>Failing Strong/Weak parts</td>
<td>26</td>
</tr>
<tr>
<td>rule-of-2</td>
<td>50</td>
</tr>
<tr>
<td>strict parts</td>
<td>92</td>
</tr>
<tr>
<td>matrices</td>
<td>227</td>
</tr>
<tr>
<td>Total</td>
<td>243</td>
</tr>
<tr>
<td>Idempotent reducts</td>
<td>32</td>
</tr>
<tr>
<td>Axioms: total</td>
<td>152</td>
</tr>
<tr>
<td>derived</td>
<td>152</td>
</tr>
<tr>
<td>Failing no altern. pc.</td>
<td>10</td>
</tr>
<tr>
<td>Alternative pc.</td>
<td>15</td>
</tr>
</tbody>
</table>
### TABLE A, for the 14 Theorems of Asset

<table>
<thead>
<tr>
<th></th>
<th>Average/form</th>
<th>Maximum</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of recursive passes</td>
<td>113</td>
<td>719</td>
<td>1578</td>
</tr>
<tr>
<td>Time (in msecs.)</td>
<td>353086</td>
<td>2845685</td>
<td>4943210</td>
</tr>
<tr>
<td>No. of nodes created</td>
<td>8188</td>
<td>58941</td>
<td>114633</td>
</tr>
<tr>
<td>linked</td>
<td>52</td>
<td>372</td>
<td>724</td>
</tr>
<tr>
<td>penalized</td>
<td>428</td>
<td>4785</td>
<td>5996</td>
</tr>
<tr>
<td>as axioms</td>
<td>159</td>
<td>1534</td>
<td>2220</td>
</tr>
<tr>
<td>derived axs.</td>
<td>399</td>
<td>3031</td>
<td>5589</td>
</tr>
<tr>
<td>pruned</td>
<td>3587</td>
<td>18704</td>
<td>50224</td>
</tr>
<tr>
<td>No. of alternative pc.</td>
<td>77</td>
<td>312</td>
<td>1077</td>
</tr>
<tr>
<td>Failed: no altern. pc.</td>
<td>10</td>
<td>60</td>
<td>146</td>
</tr>
<tr>
<td>Size of Black list</td>
<td>106</td>
<td>243</td>
<td>1481</td>
</tr>
<tr>
<td>White list</td>
<td>106</td>
<td>243</td>
<td>1481</td>
</tr>
<tr>
<td>Final Proof: nodes</td>
<td>135</td>
<td>388</td>
<td>1883</td>
</tr>
<tr>
<td>width</td>
<td>47</td>
<td>128</td>
<td>663</td>
</tr>
<tr>
<td>depth</td>
<td>29</td>
<td>83</td>
<td>406</td>
</tr>
</tbody>
</table>

### TABLE B, for the 14 Theorems of Asset

<table>
<thead>
<tr>
<th></th>
<th>No. of formulas where % reaches</th>
<th>% of all</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+1%</td>
<td>+10%</td>
</tr>
<tr>
<td>Failing Curry property</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>Failing Strong/Weak parts</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>rule-of-2</td>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>strict parts</td>
<td>14</td>
<td>7</td>
</tr>
<tr>
<td>matrices</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Total</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Idempotent reducts</td>
<td>14</td>
<td>3</td>
</tr>
<tr>
<td>Axioms: total</td>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>derived</td>
<td>14</td>
<td>2</td>
</tr>
<tr>
<td>Failing no altern. pc.</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Alternative pc.</td>
<td>11</td>
<td>0</td>
</tr>
</tbody>
</table>
TABLE A, for the 18 Incomplete Attempts of Asset

<table>
<thead>
<tr>
<th></th>
<th>Average/form</th>
<th>Maximum</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of recursive passes</td>
<td>277</td>
<td>3161</td>
<td>4988</td>
</tr>
<tr>
<td>Time (in msecs.)</td>
<td>3395209</td>
<td>3875300</td>
<td>61113755</td>
</tr>
<tr>
<td>No. of nodes created</td>
<td>85251</td>
<td>163314</td>
<td>1434520</td>
</tr>
<tr>
<td>linked</td>
<td>320</td>
<td>2136</td>
<td>5756</td>
</tr>
<tr>
<td>penalized</td>
<td>912</td>
<td>5895</td>
<td>16410</td>
</tr>
<tr>
<td>as axioms</td>
<td>476</td>
<td>1364</td>
<td>8573</td>
</tr>
<tr>
<td>derived axs.</td>
<td>1579</td>
<td>6459</td>
<td>28421</td>
</tr>
<tr>
<td>pruned</td>
<td>47394</td>
<td>107392</td>
<td>853094</td>
</tr>
<tr>
<td>No. of alternative pc.</td>
<td>582</td>
<td>1781</td>
<td>10478</td>
</tr>
<tr>
<td>Failed: no altern. pc.</td>
<td>20</td>
<td>122</td>
<td>367</td>
</tr>
<tr>
<td>Size of Black list</td>
<td>19</td>
<td>100</td>
<td>339</td>
</tr>
<tr>
<td>White list</td>
<td>72</td>
<td>189</td>
<td>1298</td>
</tr>
</tbody>
</table>

TABLE B, for the 18 Incomplete Attempts of Asset

<table>
<thead>
<tr>
<th>No. of formulas where % reaches</th>
<th>+1%</th>
<th>+10%</th>
<th>+20%</th>
<th>+50%</th>
<th>nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failing Curry property</td>
<td>18</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>Failing Strong/Weak parts</td>
<td>15</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>rule-of-2</td>
<td>11</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>strict parts</td>
<td>18</td>
<td>12</td>
<td>6</td>
<td>1</td>
<td>17</td>
</tr>
<tr>
<td>matrices</td>
<td>18</td>
<td>16</td>
<td>14</td>
<td>2</td>
<td>28</td>
</tr>
<tr>
<td>Total</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>11</td>
<td>56</td>
</tr>
<tr>
<td>Idempotent reducts</td>
<td>18</td>
<td>8</td>
<td>1</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Axioms: total</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>derived</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Failing no altern. pc.</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alternative pc.</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
### TABLE A, for the 32 Formulas of Asset

<table>
<thead>
<tr>
<th></th>
<th>Average/form</th>
<th>Maximum</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of recursive passes</td>
<td>205</td>
<td>3161</td>
<td>6566</td>
</tr>
<tr>
<td>Time (in msecs.)</td>
<td>2064280</td>
<td>3875300</td>
<td>66056965</td>
</tr>
<tr>
<td>No. of nodes created</td>
<td>51536</td>
<td>163314</td>
<td>1649153</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>linked</td>
<td>203</td>
<td>2136</td>
<td>6480</td>
</tr>
<tr>
<td>penalized</td>
<td>700</td>
<td>5895</td>
<td>22406</td>
</tr>
<tr>
<td>as axioms</td>
<td>337</td>
<td>1534</td>
<td>10793</td>
</tr>
<tr>
<td>derived axs.</td>
<td>1063</td>
<td>6459</td>
<td>34010</td>
</tr>
<tr>
<td>pruned</td>
<td>28229</td>
<td>107392</td>
<td>903319</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. of alternative pc.</td>
<td>361</td>
<td>1781</td>
<td>11555</td>
</tr>
<tr>
<td>Failed: no altern. pc.</td>
<td>16</td>
<td>122</td>
<td>513</td>
</tr>
<tr>
<td>Size of Black list</td>
<td>15</td>
<td>100</td>
<td>476</td>
</tr>
<tr>
<td>White list</td>
<td>87</td>
<td>243</td>
<td>2779</td>
</tr>
</tbody>
</table>

### TABLE B, for the 32 Formulas of Asset

<table>
<thead>
<tr>
<th></th>
<th>No. of formulas where % reaches</th>
<th>% of all</th>
<th>nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+1%</td>
<td>+10%</td>
<td>+20%</td>
</tr>
<tr>
<td>Failing Curry property</td>
<td>30</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>Failing Strong/Weak parts</td>
<td>23</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>rule-of-2</td>
<td>24</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>strict parts</td>
<td>32</td>
<td>19</td>
<td>9</td>
</tr>
<tr>
<td>matrices</td>
<td>32</td>
<td>30</td>
<td>26</td>
</tr>
<tr>
<td>Total</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>Idempotent reducts</td>
<td>32</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>Axioms: total</td>
<td>32</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>derived</td>
<td>32</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Failing no altern. pc.</td>
<td>21</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alternative pc.</td>
<td>29</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Appendix B
Performance on Associative Formulas

KRIPKE's performance on the statements of associativity of the formulas in Table 1-1 is summarized under the tables for Asset in Appendix A, but we give here some information concerning KRIPKE's performance on these statements considered individually. For each associative statement, defined by the formulas 1-16 of Table 1-1, we give information for each direction of associativity; i.e. for $1 \leq i \leq 16$, we test each of the following, with one being the left-to-right direction, and the other the right-to-left direction

$$(C \Phi_i (D \Phi_i E)) \rightarrow ((C \Phi_i D) \Phi_i E)$$

$$(C \Phi_i D) \Phi_i E) \rightarrow (C \Phi_i (D \Phi_i E))$$

Where a direction is provable, we give the CPU time required to prove it, in seconds, and the size of the final proof. We warn the reader that the size given may be somewhat misleading in that it does not include nodes that belong to repeated instances of derived axioms, linked subtrees, and the like. Incomplete directions are those that KRIPKE failed to terminate on within one hour of CPU time; we give the size of the proof search tree when KRIPKE abandoned its search.
<table>
<thead>
<tr>
<th>No.</th>
<th>Status</th>
<th>Time</th>
<th>Size</th>
<th>Status</th>
<th>Time</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>theorem</td>
<td>75</td>
<td>87</td>
<td>incomplete</td>
<td>5000</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>theorem</td>
<td>5</td>
<td>51</td>
<td>incomplete</td>
<td>1040</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>incomplete</td>
<td>432</td>
<td></td>
<td>theorem</td>
<td>10</td>
<td>58</td>
</tr>
<tr>
<td>4</td>
<td>theorem</td>
<td>98</td>
<td>88</td>
<td>incomplete</td>
<td>859</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>theorem</td>
<td>109</td>
<td>146</td>
<td>incomplete</td>
<td>1452</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>incomplete</td>
<td>122</td>
<td></td>
<td>incomplete</td>
<td>581</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>incomplete</td>
<td>4824</td>
<td></td>
<td>theorem</td>
<td>807</td>
<td>309</td>
</tr>
<tr>
<td>8</td>
<td>theorem</td>
<td>332</td>
<td>97</td>
<td>incomplete</td>
<td>618</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>theorem</td>
<td>110</td>
<td>90</td>
<td>theorem</td>
<td>2845</td>
<td>87</td>
</tr>
<tr>
<td>10</td>
<td>theorem</td>
<td>35</td>
<td>120</td>
<td>incomplete</td>
<td>2094</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>incomplete</td>
<td>489</td>
<td></td>
<td>incomplete</td>
<td>900</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>incomplete</td>
<td>3121</td>
<td></td>
<td>incomplete</td>
<td>3210</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>incomplete</td>
<td>632</td>
<td></td>
<td>theorem</td>
<td>312</td>
<td>388</td>
</tr>
<tr>
<td>14</td>
<td>incomplete</td>
<td>4893</td>
<td></td>
<td>incomplete</td>
<td>4669</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>incomplete</td>
<td>2078</td>
<td></td>
<td>theorem</td>
<td>102</td>
<td>142</td>
</tr>
<tr>
<td>16</td>
<td>theorem</td>
<td>20</td>
<td>78</td>
<td>theorem</td>
<td>71</td>
<td>142</td>
</tr>
</tbody>
</table>
References

[Anderson and Belnap 75]
*Entailment: The Logic of Relevance and Necessity, Vol.1.*

[Belnap 77]
Belnap, N.D. Jr.
How a Computer Should Think.

[Belnap 78]
Belnap, N.D. Jr.
A Useful Four-Valued Logic.

[Belnap and Dunn 81]
Belnap, N.D. Jr. and Dunn, J.M.
Entailment and the Disjunctive Syllogism.

[Belnap and Wallace 61]
Belnap, N.D. Jr. and Wallace, J.R.
*A Decision Procedure for the System E7 of Entailment with Negation.*

[Belnap and Wallace 65]
Belnap, N.D. Jr. and Wallace, J.R.
A Decision Procedure for the System E7 of Entailment with Negation.

[Bibel 77]
Bibel, W.
A Syntactic Connection Between Proof Procedures and Refutation Procedures.
[Bibel 80] Bibel, W.
A Theoretical Basis for Systematic Proof Procedure.

[Bibel 82] Bibel, W.
_Automated Theorem Proving._
Vieweg, Braunschweig, 1982.

[Bibel and Schreiber 75]
Bibel, W. and Schreiber, J.
Proof Search in A Gentzen-Like System of First Order Logic.
In _Proceedings of the International Computing Symposium_,

[Blasius et. al. 81]
Blasius, K., Eisenger, N., Siekmann, J., Smolka, G., Herold, A. and Walther, C.
The Markgraf Karl Refutation Procedure (Fall 1981).
Interne Bericht, University of Karlsruhe.
1981.

[Bledsoe 77] Bledsoe, W.W.
Non-Resolution Theorem Proving.

[Bunder 82] Bunder, M.
_Order of Strength Relations between Formal Systems with Applications to Metamathematics._

[Church 51] Church, A.
The Weak Theory of Implication.
In Menne, Wilhem and Angsil (eds.), _Kontrolliertes Denken,
Untersuchungen zum Logikkalkül und zur Logik der Einzelwissenschaften_, 22-37.
Kommissions-Verlag Karl Alber, Munich, 1951.

[Church 56] Church, A.
_Introduction to Mathematical Logic._
6th printing, 1970.
[Cialdea 83]
Cialdea, M.
Une Methode de Resolution pour la Logique Intuitionniste Propositionnelle - via un Systeme Modal.
1983.

[Clocksin and Mellish 81]
Clocksin, W.F. and Mellish, C.S.
Programming in Prolog.

[Copeland 79]
Copeland, B.J.
On When a Semantics is Not a Semantics: Some Reasons for Disliking the Routley-Meyer Semantics for Relevance Logic.

[Curry 50]
Curry, H.B.
A Theory of Formal Deducibility.
Ann Arbor, Michigan, 1950.

[Curry 63]
Curry, H.B.
Foundations of Mathematical Logic.

[Davis and Putnam 60]
Davis, M. and Putnam, H.
A Computing Procedure for Quantification Theory.

[Dershowitz and Manna 79]
Dershowitz, N. and Manna, Z.
Proving Terminations with Multiset Orderings.

[Dickson 13]
Dickson, L.E.
Finiteness of the Odd Perfect and Primitive Abundant Numbers with n Distinct Prime Factors.
[Doyle 78] Doyle, J.
Truth Maintenance Systems for Problem Solving.

[Dummett 77] Dummett, M.
Elements of Intuitionism.

[Dunn 66] Dunn, J.M.
The Algebra of Intensional Logics.

[Dunn 75] Dunn, J.M.
The Algebra of R.

[Dunn 82] Dunn, J.M.
Quantum Mathematics.

[Dunn 8?] Dunn, J.M.
Relevance Logic and Entailment.
In Gabbay, D. and Guenthner, F. (eds.), Handbook of Philosophical Logic, Reidel, Dordrecht, Holland, 198?

[Farinas del Cerro 81] Farinas del Cerro, L.
Deduction Automatique et Logique Modale.

[Ferrante and Rackoff 79] Ferrante, J. and Rackoff, C.W.
The Computational Complexity of Logical Theories.

Implementation and Results of a Proof Procedure for First-Order Modal Logic.
Interner Bericht, Institut für Informatik, University of Karlsruhe. 1981.
Gabbay, D.  
Intuitionistic Basis for Non-Monotonic Logic.  

Gallaire, H. and Minker, J. (eds.).  
*Logic and Data Bases.*  

Galvin, F.  
Horn Sentences.  

Garey, M.R. and Johnson, D.S.  

Gentzen, G.  
Untersuchungen über das logische Schliessen.  

Grätzer, G.  
*Universal Algebra.*  
2nd. edition.

Harrop, R.  
Some Structure Results for Propositional Calculi.  

Haspel, C.H.  
*A Study of Some Interpretations of Modal and Intuitionistic Logics in the First Order Predicate Calculus.*  

Hickman, J.L.  
A Note on the Concept of Multiset.  
[Israel 80] Israel, D.J.
What's Wrong with Non-Monotonic Logic?

[Jensen and Wirth 74]
Jensen, K. and Wirth, N.

[Kalicki 52a] Kalicki, J.
A Test for the Equality of Truth-Tables.

[Kalicki 52b] Kalicki, J.
On Comparison of Finite Algebras.

[Kalicki 54] Kalicki, J.
On Equivalent Truth-Tables for Many-Valued Logics.

[Kleene 52] Kleene, S.C.
Permutability of Inferences in Gentzen's Calculi, LK and LJ.

[König 36] König, D.
*Theorie der endlichen und unendlichen Graphen; Kombinatorische Topologie der Streckenkomplexe.*
Akademische Verlagsgesellschaft, Leipzig, 1936.

[Kripke 59] Kripke, S.A.
The Problem of Entailment.
Abstract.
[Kron, Maric and Vujosevic 81]
Kron, A., Maric, Z. and Vujosevic, S.
Entailment and Quantum Logic.

[Ladner 77]
Ladner, R.E.
The Computational Complexity of Provability in Systems of Modal Propositional Logic.

[Lausch and Nöbauer 73]
Lausch, H. and Nöbauer, W.
Algebra of Polynomials.

[Lauth 83]
Lauth, E.
Une Methode de Resolution Lineaire Ordonnee, pour la Logique Temporelle Lineaire.
1983.

[Lewis 18]
Lewis, C.I.
A Survey of Symbolic Logic.
University of California, Berkeley, 1918.
Reprinted 1961, Dover.

[Lukasiewicz and Tarski 30]
Lukasiewicz, J. and Tarski, A.
Untersuchungen über den Aussagenkalkul.

[McDermott and Doyle 80]
McDermott, D. and Doyle, J.
Non-Monotonic Logic I.

[McRobbie 79]
McRobbie, M.A.
A Proof Theoretic Investigation of Relevant and Modal Logics.
[McRobbie and Belnap 79]
McRobbie, M.A. and Belnap, N.D.Jr.
Relevant Analytic Tableaux.

[McRobbie and Belnap ??]
McRobbie, M.A. and Belnap, N.D. Jr.
Proof Tableau Formulations of Some First-Order Relevant Ortho-
Logics.
Forthcoming, *Bulletin of the Section of Logic, Polish Academy of
Sciences*.

[McRobbie, Meyer and Thistlewaite 83]
McRobbie, M.A., Meyer, R.K. and Thistlewaite, P.B.
Computer-Aided Investigations into the Decision Problem for
Relevant Logics: The Search for a Free Associative
Connective.
In Goldschlager, L.M. (ed.), *Australian Computer Science
Communications*, 236-267. Basser Dept. of Computer
Science, University of Sydney, 1983.
Proceedings of the 6th Australian Computer Science Conference.

[McRobbie, Thistlewaite and Meyer 82]
McRobbie, M.A., Thistlewaite, P.B. and Meyer, R.K.
A Mechanized Decision Procedure for Non-Classical Logics: The
Program KRIPKE.
Abstract.

[Meyer 66]
Meyer, R.K.
*Topics in Modal and Many-Valued Logic.*

[Meyer 72]
Meyer, R.K.
Conservative Extension in Relevant Implication.

[Meyer 73]
Meyer, R.K.
Improved Decision Procedures for Pure Relevant Logics.
Typescript.

[Meyer 75a]
Meyer, R.K.
Arithmetic Formulated Relevantly.
Typescript.
1975.
The Consistency of Arithmetic.
Typescript.
1975.

Relevant Arithmetic.
*Bulletin of the Section of Logic, Polish Academy of the Sciences*
Abstract.

*Sentential Constants in R.*
Research Paper 2, Logic Group, Research School of Social Sciences, Australian National University, Canberra, 1977.

*A Boolean-Valued Semantics for R.*

Remarks and Results in the Relevant Theory of Propositions and Propositional Quantification.
Typescript.
1978.

The Relevant Theory of Propositions is Undecidable (with Applications to Some First-Order Theories).
Typescript.
1978.

Almost Skolem Forms for Relevant (and other) Logics.

E, R and γ.
[Meyer and McRobbie 82]
Meyer, R.K. and McRobbie, M.A.
Multisets and Relevant Implication I and II.

[Meyer and Mortensen 82]
Meyer, R.K. and Mortensen, C.
Inconsistent Models for Relevant Arithmetic.
Typescript.
1982.

[Meyer and Routley 72]
Meyer, R.K. and Routley, R.
Algebraic Analysis of Entailment I.

[Meyer and Routley 73]
Meyer, R.K. and Routley, R.
An Undecidable Relevant Logic.

[Meyer, Dunn and Leblanc 74]
Meyer, R.K., Dunn, J.M. and Leblanc, H.
Completeness of Relevant Quantification Theories.

[Meyer, Giambrone and Brady 8?]
Meyer, R.K., Giambrone, S. and Brady, R.
Where Gamma Fails.
Forthcoming, _Studia Logica._
198?

[Mitani 78]
Mitani, S.
Completion Theory Performed on the Basis of LJ.

[Mogilevski and Ostrouhov 78]
Mogilevski, G.L. and Ostrouhov, D.A.
On a Mechanical Propositional Calculus Using Smullyan's Analytic Tableau.
[Monro 83] Monro, G.P.
The Concept of Multiset.
Typescript. Department of Pure Mathematics, University of Sydney.
1983.

[Morgan 76a] Morgan, C.G.
Methods for Automated Theorem Proving in Nonclassical Logics.

[Morgan 76b] Morgan, C.
A Resolution Principle for a Class of Many-Valued Logics.

[Murray 82] Murray, N.V.
Completely Non-Clausal Theorem Proving.

[Nilsson 80] Nilsson, N.J.
*Principles of Artificial Intelligence*.

[Ohlbach and Wrightson 83a] Ohlbach, H. and Wrightson, G.
Belnap's Problem in Relevance Logic.
*Association of Automated Reasoning Newsletter* 2:3-5, 1983.

[Ohlbach and Wrightson 83b] Ohlbach, H. and Wrightson, G.
*Solving a Problem in Relevance Logic with an Automated Theorem Prover*.

[Plaisted 80] Plaisted, D.A.
An Efficient Relevance Criterion for Mechanical Theorem Proving.

[Pnueli 78] Pnueli, A.
The Temporal Semantics of Concurrent Programs.
183

[Popper 63] Popper, K.
*Conjectures and Refutations.*

[Popplestone 67] Popplestone, R.J.
Beth-Tree Methods in Automated Theorem Proving.

[Post 47] Post, E.L.
Recursive Unsolvability of a Problem of Thue.

[Pratt 77] Pratt, V.R.
A Practical Decision Method for Propositional Dynamic Logic.

[Quine 51] Quine, W.V.O.
*Mathematical Logic.*
Revised Edition.

[Rasiowa 74] Rasiowa, H.
*An Algebraic Approach to Non-Classical Logics.*

[Rasiowa and Sikorski 68] Rasiowa, H. and Sikorski, R.
*The Mathematics of Metamathematics.*
2nd edition.

[Read 80] Read, S.
How to Make Logic Relevant.

[Reiter 76] Reiter, R.
A Semantically Guided Deductive System for Automatic Theorem Proving.
[Rieger, Rosenberg and Samet 79]
Rieger, C., Rosenberg, J. and Samet, H.
Artificial Intelligence Programming Languages for Computer
Aided Manufacturing.
IEEE Transactions on Systems, Man, and Cybernetics

[Routley and Meyer 73]
Routley, R. and Meyer, R.K.
The Semantics of Entailment I.
In Leblanc, H. (ed.), Truth, Syntax and Modality, North Holland,

[Routley, Meyer, Plumwood and Brady 84]
Routley, R., Meyer, R.K., Plumwood, V. and Brady, R.
Relevant Logics and Their Rivals. Part 1: The Basic
Philosophical and Semantical Theory.
Ridgeview, Atascadero, California, 1984.

[Routley, Routley, Meyer and Martin 82]
Routley, R., Routley, V., Meyer, R.K. and Martin, E.P.
On the Philosophical Bases of Relevant Logic Semantics.

[Sakai and Miyachi 84]
Sakai, K. and Miyachi, T.
Incorporating Naive Negation into Prolog.
In Crossley, J.N. and Jaffar, J. (ed.), Logic and Computation:
Proceedings of a Conference, Monash University, Clayton,
Victoria, 1984.

[Sandford 80]
Sandford, M.M.
Using Sophisticated Models in Resolution Theorem Proving.

[Shapiro and Wand 76]
Shapiro, S.C. and Wand, M.
The Relevance of Relevance.
Technical Report 46, Computer Science Dept., Indiana
University, Bloomington, Indiana, 1976.

[Shimada 74]
Shimada, K.
A Theorem-Prover for Intuitionistic Propositional Logic.
(In Japanese).
Siekmann, J. and Wrightson, G.
Automation of Reasoning.

Siklossy, L., Rich, A. and Marinov, V.
Breadth-First Search: Some Surprising Results.

Slaney, J.K.
Computers and Relevant Logics: A Project in Computing
Matrix Model Structures for Propositional Logics.

Slaney, J.K.
3088 Varieties.

Statman, R.
Intuitionistic Propositional Logic is Polynomial Space Complete.

Tokarz, M.
Functions Definable in Sugihara Algebras and Their Fragments I.

Tokarz, M.
Functions Definable in Sugihara Algebras and Their Fragments II.

Urquhart, A.
The Undecidability of Entailment and Relevant Implication.

Urquhart, A.
Relevant Implication and Projective Geometry.
Forthcoming, Logique et Analyse.

ACM Software Engineering Notes
1980.
Contributions from the VERkshop on Formal Verification.

Weyhrauch, R.W.
Prolegomena to a Theory of Mechanized Formal Reasoning.
[Whitehead and Russell 25]
Whitehead, A.N. and Russell, B.A.W.  
*Principia Mathematica*.  
Cambridge University Press, 1925.  
3 Volumes, 2nd Ed.

[Wos 82]
Wos, L.  
Solving Open Questions with an Automated Theorem-Proving Program.  

[Wrightson 79]
Wrightson, G.  
A Proof Procedure for Higher-Order Modal Logic.  