Conditional Logic Programming

A. W. Bollen

A thesis submitted for
the degree of Doctor of Philosophy
of the Australian National University

Computer Science Department
The Faculties
December 1988
Except where otherwise acknowledged, this thesis is my own work.

[Signature]

A. W. Beller
Acknowledgments

I had the good fortune, as one whose interests tend to fall between the two stools of logic and computer science, to find at the Australian National University logicians and computer scientists striving to establish common ground between the disciplines. My thanks in particular go to Professor R. B. Stanton for his enthusiasm and sharp insight, and Dr E. P. Martin, for his interest and encouragement and for his guidance in the process of stitching together logical and computational aspects. My debts to the work of Dr R. K. Meyer and Dr R. Sylvan will be obvious. The Automated Reasoning Project, under the operational guidance of Dr M. A. McRobbie, provided a superlative research environment. Thanks also to my fellow students, in particular Rob Edmondson, Seppo Keranen and Peter Lavers, for discussion, argument and support.
Abstract

This thesis addresses the issue of extending logic programming languages (in particular PROLOG) to allow nested conditionals in assertions and queries, which implementations of PROLOG generally do not allow. Chapter 1 discusses the expressive advantages of the extension, particularly in the area of diagnosis and planning, where we will often be interested in the dual questions of ‘What would happen, were A the case?’ and ‘What assumptions would give rise to B?’.

For example, we can solve the standard circuit diagnosis problem of [Gen85] efficiently and naturally by allowing the language to express the query: ‘What sets of gate states S would imply that inputs I imply outputs O?’ At this level, the NPROLOG system of [GR84] is adequate, and the only conceptual extension to standard PROLOG concerns the handling of variables in the antecedents of conditionals. The basic method for computing a conditional query of the form ‘A→B’ at this level is to first add A to the database, and then compute B using the normal PROLOG mechanism. Such a scheme corresponds to interpreting the conditional as an intuitionistic implication. The disadvantages of this interpretation centre around the so-called ‘Positive Paradox’: whenever B follows from the database, so will A→B, for any arbitrary A. For example, in a planning system we might think to use the following definition for determining bad states and actions:

\[
\text{bad-state}(Y) \land (\text{state}(X) \rightarrow \text{state}(Y)) \rightarrow \text{bad-state}(X)
\]
with the interpretation: 'If state $X$ results in a state $Y$ which is bad, then $X$ is bad also'. But with an intuitionistic interpretation, this will lead to every state being labelled 'bad', whenever any state is. A relevant reading of the conditional is more appropriate in these circumstances. The difference is that $A \rightarrow B$ here only follows from the database when there is some real connection between $A$ and $B$. Computationally, $A \rightarrow B$ succeeds from the database only if $B$ succeeds from the database augmented with $A$ in a way which actually uses $A$. On a more abstract level, the intuitionistic computation corresponds to a 'black box' model in which we attempt to deduce the hidden internal connection structure of some situation by manipulating the elements under our control (the 'inputs') and observing the effects on the part of the environment accessible to us (the 'outputs'). In the general case this experimental method is incomplete, in the sense that we will not be able to deduce the full structure. On the other hand, the 'black box' model is inappropriate for the contexts in question, for which the structure is given explicitly by the rules in the database. A better model is one in which we determine whether or not $A$ and $B$ are connected in the appropriate sense by attempting to find a pathway from one to the other, in terms of the explicitly given connections. This model corresponds to the relevant computation performed here.

There is no introductory text for relevant logic and for the novice the field is often quite inaccessible. In Chapter 2 I present a new formulation of the positive fragment of the major relevant logic $R$ which (I hope) will do something to remedy this situation, building up to the full system from some simple intuitions about the nature of rules. One of the advantages of the approach is that it gives a simple model for intensional conjunction and disjunction ('fusion' and 'fission') which play an important role in relevant logics but which have no very clear intuitive motivation in the standard developments. Briefly, I show how $R$ can be obtained from the weak relevant logic $DA$ by adding an enthymeme principle in the form of the entailment principle:

$$Ao(AoB \rightarrow C) \Rightarrow B \rightarrow C$$

where 'o' is 'fusion', the intensional conjunction. The advantage of this view of
Richard R. is that DA relies only on principles which are as intuitive and unobjectionable as logical principles ever are, and the enthymeme principle is a straightforward extension to these. (Interpreted, it says that whenever A and B together are sufficient for C, and A is the case, then there is a clear sense in which we would want to say that B is sufficient for C.) This construction provides formal backing for the arguments in [SMPB82] to the effect that many of the logical principles accepted by the stronger relevant logics are only enthymematically true.

Chapter 3 deals with negation. Negation in relevant logics is a paraconsistent operator, which means that arbitrary B does not follow from a contradiction A \land \neg A in these systems. In other words, the logics are inconsistency tolerating: if a theory contains a contradiction then it is not necessarily the case that the theory contains everything. Paraconsistent systems have been proposed as the logical basis for expert systems and computer reasoners in general; presumably any real-world database will be inconsistent to some extent, and it would be nice to have a logic which could handle this, localizing the effects of the anomaly. From a philosophical point of view, contradictions are seen as essentially no worse than any other falsehood, and any database will contain at least some false information. This provides a theoretical framework in which computationally expensive (and incomplete) strategies for maintaining consistency in databases are optional; the system does not become logically incoherent if we fail to resolve any or all contradictions. This interest in paraconsistency in the AI world is largely a result of Belnap’s work in [Bel77a] and [Bel77b], with the first implementation described in [Bol85] and subsequent applications to logic programming systems in [BS87]. Both of these differ from Belnap’s work in rejecting Contraposition

\[ A \Rightarrow B \vdash \neg B \Rightarrow \neg A \]

(that is, whenever A entails B, not B entails not A) as a fundamental logical principle, and this is carried through to the development of this chapter. The reasons for rejecting Contraposition are two-fold. First, for an inconsistency tolerating negation there is a stable and plausible position on which it is just not true. Secondly, by rejecting Contraposition we are able to represent the system with negation within its positive fragment, normalizing expressions into forms
which can be handled by the negation-free part of the language. So whatever computational strategies we have developed for the positive fragment will also apply to the full logic.

In Chapter 4 I present some models for the conditional computation performed by my CLOGPROG (‘Conditional Logical Programming’) system. The first, and most general is the SLC (‘Linear condensed detachment for conditional clauses with selection rule’), a quite straightforward extension to the standard SLD model ([Llo84]). Next is an elaboration on the computation tree model of [GR84], which assumes a left-to-right goal selection rule and which gives a description very close to the Gentzen-style logical formulation later in the chapter. Last is a ‘Conditional Query Evaluation Procure’ (‘CQEP’), which further restricts the general model by placing a fixed ordering on database clauses selected to match goals, and is essentially a description of the algorithm computed by CLOGPROG. Given the usage requirement (discussed above) for the relevant conditional, these models should appear as quite straightforward extensions to the logic programming standard. From a formal logical point of view, however, the details are quite subtle. Relevant logics in general are very expensive computationally; the current pinnacle in the area is the ‘Kripke’ system of [TMM88] which is far too expensive to form the basis of a logic programming system. For practical purposes it is necessary to trade off expressive power against efficiency. I show how a syntactically ambiguous interpretation of the ‘user-level’ conjunction as fusion in query positions and ‘normal’ extensional conjunction in assertion positions fulfills the goals of minimizing the reduction in expressive power while supporting this simple extension to SLD resolution (and the PROLOG scheme in particular). I attempt some independent justifications of this ambiguous reading of conjunction and investigate some of the logical issues. I give the proofs of the soundness and completeness of the computation with respect to the unusual fragment of $R$ thus derived. The chapter concludes with a brief description of the implementation of CQEP in CLOGPROG, a simple extension to the familiar structure-sharing, stack-frame interpreter described in [Hog84].

Chapter 5 is a more speculative discussion of the counterfactual conditional, and
the issues of consistency maintenance. Given a counterfactual query $A \rightarrow B$ and a database $T$, the basic strategy is to determine whether the corresponding indicative conditional $A \rightarrow B$ follows from the part of $T$ cotenable with $A$, symbolically ‘$\text{Cot}(A, T)$’. In other words, we can only use the part of the database which is compatible (in some sense) with the antecedent of the query. This view of the counterfactual is an abstracted form of that first proposed by Goodman in [Goo83]. The object ‘$\text{Cot}(A, T)$’ generally will not be uniquely defined; there will normally be many different ways of removing information from $T$ to leave a residue of elements cotenable with $A$. I give a strategy for resolving this ambiguity (or at least reducing the choices). The first step is to identify contests between elements of $T$ (in the context of $A$); there is a contest between $C_1$ and $C_2$ if, intuitively, keeping $C_1$ will force $C_2$ to be excluded, and vice versa. Then we introduce a ranking between contesting elements, written as ‘$C_1 \sqsubseteq C_2$’, interpreted as: ‘In a contest between $C_1$ and $C_2$, $C_1$ wins’. This ranking carries no philosophical baggage; there are no assumptions made about why $C_1$ should win, such considerations being left up to the user. I introduce some computational models extending those of the previous chapter incorporating this contest resolution mechanism. There is a brief comparison with the logically more general, but computationally less well-directed development of [Gin86]. The same type of strategy is applicable to the problem of consistency maintenance, and computational models are also provided for this aspect. Finally I give a short description of the implementation of counterfactuals and consistency maintenance in CLOG-PROG.
Contents

1 Motivation

1.1 A Diagnosis Example ........................................ 12
1.2 Rules, Connections and Relevance ................................. 15
1.3 A Planning Example ................................................ 19
1.4 A Propaganda Example ........................................... 22
1.5 Abduction, Reflection and Non-Regularity ...................... 24
1.6 Relevance in Context ............................................. 25

2 A Formulation of Relevance ........................................ 30

2.1 Rules, Entailments, Theories and Logics ....................... 30
  2.1.1 Syntax of the Object Language ............................ 31
  2.1.2 Entailment Systems ........................................ 32
  2.1.3 Theories .................................................. 34
  2.1.4 Logics .................................................. 34
  2.1.5 Summary ................................................ 37
2.2 Extensional Conjunction and Disjunction ....................... 37
2.3 Intensional Conjunction and Disjunction ....................... 40
2.4 Implication .................................................. 43
CONTENTS

2.4.1 Basic Principles ............................................. 44
2.4.2 Implication with Fusion and Fission ....................... 45
2.4.3 Elimination Principles ...................................... 46
2.4.4 Enthymemes .................................................. 47
2.4.5 Enthymeme vs Residuation .................................. 49
2.5 Permission ...................................................... 51
2.6 Positive Systems with Quantifiers ............................ 55
2.7 Summary of Positive Principles .............................. 56
2.8 Hierarchy of Positive Systems ............................... 59
2.9 Some Properties of Positive Entailment Systems ............ 61
  2.9.1 Weak Relevance ............................................ 61
  2.9.2 Tautological Entailments ................................. 62
  2.9.3 Replacements .............................................. 62
  2.9.4 Reductions ................................................ 64
  2.9.5 Conservative Extensions .................................. 66
2.10 Properties of Theories and Logics .......................... 72
  2.10.1 Definitions ............................................... 72
  2.10.2 ‘Constructing’ Theories and Logics .................... 73
  2.10.3 Entailment Normality of Non-Enthymeme Logics ......... 75
  2.10.4 Entailment and Logical Implication in Enthymeme Systems 78
  2.10.5 Some Properties of Enthymeme Logics ................ 80
  2.10.6 Conservative Extensions ............................... 82
  2.10.7 Comparison With Other Systems ....................... 83
2.11 Logic-Based Theories ........................................ 88
# CONTENTS

## 3 Negation

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Setting the Agenda</td>
<td>92</td>
</tr>
<tr>
<td>3.1.1 'Positive' and 'Failure' Negations</td>
<td>92</td>
</tr>
<tr>
<td>3.1.2 Inconsistency and Paraconsistency</td>
<td>93</td>
</tr>
<tr>
<td>3.1.3 Truth and Falsity</td>
<td>96</td>
</tr>
<tr>
<td>3.1.4 Strategy for Formulating Negation Principles</td>
<td>100</td>
</tr>
<tr>
<td>3.2 Falsity Conditions for 'And' and 'Or'</td>
<td>100</td>
</tr>
<tr>
<td>3.3 Negation and Implication</td>
<td>101</td>
</tr>
<tr>
<td>3.4 Hierarchy of Systems with Negation</td>
<td>103</td>
</tr>
<tr>
<td>3.5 Some Properties of Systems with Negation</td>
<td>103</td>
</tr>
<tr>
<td>3.5.1 Negation Normal Form</td>
<td>103</td>
</tr>
<tr>
<td>3.5.2 Conservative Extensions</td>
<td>105</td>
</tr>
<tr>
<td>3.5.3 Contraposition</td>
<td>106</td>
</tr>
<tr>
<td>3.5.4 Replacements</td>
<td>108</td>
</tr>
<tr>
<td>3.6 Some Properties of Logics with Negation</td>
<td>110</td>
</tr>
<tr>
<td>3.7 Discussion</td>
<td>111</td>
</tr>
<tr>
<td>3.7.1 Negation in Standard Relevant Logics</td>
<td>111</td>
</tr>
<tr>
<td>3.7.2 Contraposition Via Suppression</td>
<td>117</td>
</tr>
<tr>
<td>3.7.3 Rejecting Contraposed Forms of Extended Principles</td>
<td>119</td>
</tr>
<tr>
<td>3.7.4 Contraposition Via Alternative Falsity Conditions for Implication</td>
<td>120</td>
</tr>
<tr>
<td>3.7.5 Possible Failure of Replacement Principles</td>
<td>122</td>
</tr>
<tr>
<td>3.8 Suppression Systems</td>
<td>123</td>
</tr>
</tbody>
</table>
## CONTENTS

### 4 Relevant Logic Programming  

<table>
<thead>
<tr>
<th>4.1 Introduction</th>
<th>4.2 The Computation</th>
<th>4.3 The Logic</th>
<th>4.4 Conjunction Revisited</th>
<th>4.5 Comparing the Logic and the Computation</th>
<th>4.6 Implementation</th>
<th>4.7 Speculations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>127</td>
<td>129</td>
<td>153</td>
<td>182</td>
<td>185</td>
<td>203</td>
<td>205</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4.2.1 Syntax</th>
<th>4.2.2 Substitutions</th>
<th>4.2.3 Goal of the Computation</th>
<th>4.2.4 The General Model</th>
<th>4.2.5 The Computation Tree Model</th>
<th>4.2.6 Comparing SLC Proofs and Computation Trees</th>
<th>4.2.7 A Conditional Query Evaluation Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>129</td>
<td>131</td>
<td>131</td>
<td>131</td>
<td>138</td>
<td>144</td>
<td>145</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4.2.8 Substitutions</th>
<th>4.2.9 Soundness of the Computation</th>
<th>4.2.10 Completeness of the Computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>131</td>
<td>188</td>
<td>192</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4.3.1 The Logical Framework</th>
<th>4.3.2 Gentzen Formulations</th>
<th>4.3.3 Towards a Computationally Viable Fragment</th>
<th>4.3.4 Formalizing the Fragment</th>
<th>4.3.5 Restoring Negation</th>
</tr>
</thead>
<tbody>
<tr>
<td>153</td>
<td>156</td>
<td>160</td>
<td>165</td>
<td>177</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4.4.1 Overview</th>
<th>4.4.2 Substitutions</th>
<th>4.4.3 Soundness of the Computation</th>
<th>4.4.4 Completeness of the Computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>185</td>
<td>186</td>
<td>188</td>
<td>192</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4.5.1 Overview</th>
<th>4.5.2 Substitutions</th>
<th>4.5.3 Soundness of the Computation</th>
<th>4.5.4 Completeness of the Computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>185</td>
<td>186</td>
<td>188</td>
<td>192</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4.6 Implementation</th>
<th>4.7 Speculations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>203</td>
<td>205</td>
</tr>
</tbody>
</table>
CONTENTS

4.7.1 Loop Detection ........................................... 205
4.7.2 Negation as Failure ..................................... 207

5 Counterfactuals ............................................. 208

5.1 Introduction ................................................. 208
5.2 Syntax .......................................................... 208
5.3 Cotenability ..................................................... 210
5.4 Outline of the Strategy ..................................... 212
5.5 Contest Divisions ............................................. 213
5.6 The Computation .............................................. 222
  5.6.1 Algorithmic Description ............................... 223
  5.6.2 The SLCC Model ......................................... 226
5.7 Generalizing the Model ..................................... 229
5.8 First Order Considerations ................................ 231
5.9 Consistency Maintenance ................................... 232
5.10 Discussion ..................................................... 234
5.11 Implementation .............................................. 236
Chapter 1

Motivation

The reading of a conditional $A \rightarrow B$ is very generally 'Were $A$ true, would $B$ be true?'. The Ramsey Test provides a simple device for describing different approaches to conditionals; in abstract terms it looks like this:

**Definition 1 (Ramsey Test)** $T \vdash A \rightarrow B$ iff $T + A \vdash B$

That is, $A \rightarrow B$ follows from $T$ iff $B$ follows from $T$ with $A$ somehow added to it. Different conditionals will result from different ways of formalizing the notion 'follows from' and the notion of 'database addition'.

The first point to note is that we are in different territory here than if, say, we were to read $A \rightarrow B$ as a simple material implication, $A \supset B$, with the (problematic) standard logic programming translation $-A \lor B$ (where ‘$-$’ is negation-as-failure). For example, assume that $T \not\vdash A$, $T \not\vdash B$ and furthermore $T + A \not\vdash B$. Then $T \vdash A \supset B$ (since $T \vdash \neg A$), but this is not the desired result; the query ‘Were $A$ true, would $B$ be true?’ should fail, since $B$ continues to fail when we assume $A$. Conditionals cannot be expressed in the logical language of normal PROLOG (even if it is generalized along the lines of Lloyd and Topor's [LT84],[LT85], say); certainly we would expect it to be the case that meta-interpreters for whatever formal systems for conditionals we devise could be embedded in PROLOG, but this is a different question. On the other hand, conditionals are properly a part of logic, perhaps the most important part, and are candidates in good standing...
for direct inclusion in a logic programming language. Logic is more than just the (classical) first-order predicate calculus; historically and practically it is primarily concerned with arguments, and it is this aspect that we would hope to capture with conditionals. The query $A \rightarrow B$ equates to the question, ‘Is there a good argument for $B$ on the assumption of $A$?’ which we may take as the fundamental question addressed by logic. We can only gain by allowing our languages to express such questions directly.

Natural applications for conditionals would seem to lie in the area of planning and diagnosis, where we will be interested in what follows on the assumption of some hypotheses (‘For what $B$ does $A \rightarrow B$ hold?’) and the dual question of what assumptions would produce some specified conclusion (‘For what $A$ does $A \rightarrow B$ hold?’). Many, if not most, such applications will be counterfactual in nature (‘What would happen if we rezoned suburb A to Commercial?’); in this case the ‘$+$’ operation in our statement of the Ramsey Test must include a mechanism for removing elements of $T$ incompatible (in some sense) with $A$. Until Chapter 5, we restrict ourselves to the indicative conditional, in which the ‘$+$’ operation is purely incremental; the existing information in $T$ is not revised to accommodate the new $A$. As will become evident, most of the issues we discuss here are common to the two types, and must be dealt with before the more difficult considerations special to counterfactuals.

Gabbay and Reyle ([GR84]) address the issues of extending the standard PROLOG base with indicative conditionals in the form of intuitionistic implications, in their ‘NPROLOG’ system. Here we will focus more on relevant implications, which have the advantage of being able to express the notion of a rule; we will attempt to show why this is a desirable feature. In computational terms, a relevant $A \rightarrow B$ succeeds from $T$ only if $B$ succeeds from $T + A$ in a way which actually uses $A$. In Chapter 4 it will become clear that the computation performed by our relevant ‘CLOGPROG’ (‘Conditional Logic Programming’) system is a straightforward generalization of NPROLOG; CLOGPROG can be made to behave like NPROLOG simply by disabling the machinery for checking usage, and the implementation includes a switch RELEVANT/NORELEVANT for accomplishing
1.1 A Diagnosis Example

Our first example is intended to motivate conditional logic programming in general, and give some sort of feel for what it might be like to program in such a system. The example we use is the by-now standard circuit diagnosis problem from [Gen85], more particularly with reference to the version found in [Poo86, p9]. In [Gin86], Ginsberg discusses the application of counterfactuals to this problem, but as we show here there is a sensible solution which uses only the resources of the indicative conditional. Consider the full adder circuit of Fig. 1.1. The circuit is defined formally as a series of facts stating the types of the gates and their connections:

- \( \text{gate}(D, T) \) - device \( D \) is a gate which is supposed to be of type \( T \). So we have \( \text{gate}(\text{x1}, \text{xor}) \), for example.

- \( \text{conn}(W_1, W_2) \) - input or output \( W_1 \) is connected by a wire to input or output \( W_2 \). Inputs and outputs are denoted by \( \text{in}(N, D) \) and \( \text{out}(N, D) \) where \( D \) is a device and \( N \) is a number indicating the \( N \)-th input or output port for that device. For example, \( \text{conn}(\text{out}(1, a2), \text{in}(1, o1)) \).

The total circuit is called \( f1 \). The intended operation of gates is modelled by rules like the following for OR gates:

\[
\text{val}(\text{in}(1, Gate), \text{on}) \Rightarrow \text{out-val}(\text{or}, Gate, \text{on})
\]

\[
\text{val}(\text{in}(2, Gate), \text{on}) \Rightarrow \text{out-val}(\text{or}, Gate, \text{on})
\]

\[
\text{val}(\text{in}(1, Gate), \text{off}) \land \text{val}(\text{in}(2, Gate), \text{off}) \Rightarrow \text{out-val}(\text{or}, Gate, \text{off})
\]

and the appropriate rules for AND and XOR gates. Gates may be either ok, or bad in which case they produce the opposite to the expected output for a given input:
We need a rule giving the effect of the connections:

\[
\text{conn}(X, Y) \land \text{val}(X, Z) \rightarrow \text{val}(Y, Z)
\]

For convenience we will take the following input and output rules:

\[
\text{input}(i(I_1, I_2, I_3)) \rightarrow \text{val}(\text{in}(1, f_1), I_1)
\]

\[
\text{input}(i(I_1, I_2, I_3)) \rightarrow \text{val}(\text{in}(2, f_1), I_2)
\]

\[
\text{input}(i(I_1, I_2, I_3)) \rightarrow \text{val}(\text{in}(3, f_1), I_3)
\]

\[
\text{val}(\text{out}(1, f_1), O_1) \land \text{val}(\text{out}(2, f_1), O_2) \rightarrow \text{output}(o(O_1, O_2))
\]

A diagnosis for this circuit might be a list of the possible states ('ok' or 'bad') for all the gates which contribute to a given input/output relationship. We can represent such a diagnosis by the following rule:

\[(\text{state-set}(S_s) \rightarrow (\text{input}(I) \rightarrow \text{output}(O))) \rightarrow \text{diagnosis}(I, O, S_s)\]
state-set(Ss) groups together a collection of gate states; see below. A call to 
diagnosis(Input, Output, States) with Input and Output fully instantiated will 
return a list of the gate states which implies that the given values for Input 
imply the given values for Output. To use the simplest form of the Ramsey 
Test (in fact that employed by the NPROLOG of [GR84]), to answer the query 
diagnosis(I, O, S) we first of all add state-set(S) to the database, followed by 
input(I), and then compute the query output(O) from the expanded database in 
the normal way. The only conceptual extension to normal PROLOG involved (as 
[GR84] point out) is a mechanism for ensuring that the instantiations found for 
the variables in these added elements (I and Ss) also apply to the corresponding 
variables in the answer, because unlike the normal case for assertions, they are 
not to be taken as universally quantified when added to the database. Note the 
generality of this solution; the diagnosis predicate as defined is suitable for any 
circuit description of this type.

state-set can be implemented using the lookup predicate given in [SS86], p250:

\[ !\rightarrow \text{lookup}(Gate, [[Gate, St]|Ss], St) \]

\[ Gate_1 \neq Gate_2 \land \text{lookup}(Gate_1, Ss, St) \rightarrow \text{lookup}(Gate_1, [[Gate_2, ...]|Ss], St) \]

\[ \text{state-set}(Ss) \land \text{lookup}(Gate, Ss, St) \rightarrow \text{state}(Gate, St) \]

Notice the way state-set is intended to behave as an assertion. Every time 
we call state(Gate, State) (with Gate and State instantiated) in the course of 
finding ways for input(I)\rightarrow output(O) to hold, we look up the state of that gate 
in Ss. If no entry exists for it, we add one, and if there is a contradictory entry 
we backtrack. So Ss ends up containing state entries for every gate involved in 
some path through the circuit which allows the relation input(I)\rightarrow output(O) 
to hold.

This computation is quite efficient and with the correct handling of variables 
produces the same set of diagnoses as the Theorist-based system of [Poo86]. For 
the example given there (p12), the call is
diagnosis(i(off, off, off), o(on, on), States)

with the following six combinations of possible gate states produced:

<table>
<thead>
<tr>
<th></th>
<th>a1</th>
<th>a2</th>
<th>o1</th>
<th>x1</th>
<th>x2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>bad</td>
<td>ok</td>
<td>bad</td>
<td>ok</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>bad</td>
<td>ok</td>
<td>ok</td>
<td>bad</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>bad</td>
<td>ok</td>
<td>bad</td>
<td>ok</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>bad</td>
<td>ok</td>
<td>ok</td>
<td>bad</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>ok</td>
<td>ok</td>
<td>bad</td>
<td>bad</td>
<td>ok</td>
</tr>
<tr>
<td>6</td>
<td>ok</td>
<td>ok</td>
<td>bad</td>
<td>ok</td>
<td>bad</td>
</tr>
</tbody>
</table>

In the table, a blank entry indicates that the state of that gate is irrelevant to the diagnosis; for example, in the path through the circuit represented by diagnosis 1, no call to state(a1, State) was made.

1.2 Rules, Connections and Relevance

In the circuit example of the previous section the connections between gates were given explicitly via the conn predicate. This is appropriate for a circuit diagnosis application (where we might want to extend the problem to consider possible failure in the connections themselves). But let us abstract to the general situation for which we have just a body of rules and facts describing some world, and where we may very well want to have the same ability to decide what is connected to what. In terms of the circuit example, instead of representing the connections by statements like \( \text{conn}(\text{in}(1, f1), \text{in}(1, x1)) \), we use a rule of the form \( \text{val}(\text{in}(1, f1), V) \rightarrow \text{val}(\text{in}(1, x1), V) \); we no longer need our rule \( \text{val}(X, Y) \land \text{conn}(X, Z) \rightarrow \text{val}(Z, Y) \). To verbalize the difference in emphasis, for simple circuit examples and the like we think of the rule-structure ('If \( X \) has value \( V_1 \) then \( Y \) has value \( V_2 \)') of the situation as being determined by the connection-structure ('\( X \) is connected to \( Y \)'), while for less tightly constrained problems the situation is reversed; we start off with a loose body of rules and will often be interested in the connections they generate.
In this example we can easily recover the connection information, by defining a `conn` predicate as follows:

\[
\text{port}(\text{in}(\_, \_))
\]

\[
\text{port}(\text{out}(\_, \_))
\]

\[
\text{port}(P_1) \land \text{port}(P_2) \land (\text{val}(P_1, V) \rightarrow \text{val}(P_2, V)) \rightarrow \text{conn}(P_1, P_2)
\]

A call to `conn(P_1, P_2)` will return all of the connection information of our original formulation (plus the information that everything is connected to itself, which need not worry us here). But the circuit example is very tightly constrained; in particular, the value (on or off) at any point is determined completely by the inputs and the states of the gates. More realistic problems (abstracting away from circuits to general descriptions of worlds) are rarely presented to us in this neat form.

So let us take a more general example; say a collection of light bulbs, some on and some off. The behaviour of the system is partly determined by rules like 'If $A_1$ and $A_2$ are on, then $B$ is on' and so forth, but some lights are just on for no particular reason (they are 'facts'). Take two bulbs, $A$ and $B$, which are both on, and ask the question: 'Is $B$ on because $A$ is on?'. Side-stepping the issue of exactly what we mean by 'because', perhaps our first thought would be to answer this question in the same way that we recovered the `conn` information in the circuit example, by phrasing it as $A \rightarrow B$ - 'Does $A$ (being on) imply $B$ (is on)'. In other words, determine whether or not the 'meaning' of the situation includes a rule which states that if $A$ is on, the $B$ is on. But this will not work given the simple version of the Ramsey Test we have used so far, just because the success of the query $A \rightarrow B$ will not correspond to there being a rule connecting $A$ and $B$. We add $A$ to the situation, and ask the query $B$, which will of course succeed with or without $A$, since $B$ is 'already there'. This is where the expressive advantage of the relevant conditional makes itself felt, because $A \rightarrow B$ taken relevantly does correspond very closely to the intuitive notion of a rule. In Ramsey Test terms, $A \rightarrow B$ succeeds from $T$ if and only if $B$ succeeds from $T$ in a way which actually
uses $A$. So if $A \rightarrow B$ succeeds, we know that $A$ being on has at least something to do with $B$ being on. On the other hand, $A \rightarrow B$ taken irrelevantly has the approximate force of: 'Either $B$ is on, or there is a rule connecting $A$ and $B$'. Note the asymmetry between assertions and queries in an irrelevant system; when we assert $A \rightarrow B$ we surely mean it to be taken as a rule, but the query $A \rightarrow B$ fails to express the same intention.

How would we go about answering the question in an irrelevant system, given that $A \rightarrow B$ will not work? Probably the simplest way is by the same kind of method we used in the original diagnosis example to pick up the relevant gate states. Add this rule to the situation:

$$\text{tick(usedA)} \rightarrow A$$

(where $\text{usedA}$ is a constant not used anywhere else) and ask the question

$$(\text{tick}(X) \rightarrow B) \land X =\text{usedA}$$

where $\equiv$ is an operator which tests the actual instantiations of terms for equality rather than unifying; alternatively we could use a delay mechanism with a sound equality. Either way, the solution needs at least some extra-logical facilities; either control information or the $\equiv$ operator. The question of the value of a relevant system will come down to whether we are prepared to use this kind of hack whenever we want to ask a rule-like query. Our own opinion is that $A \rightarrow B$ is generally meant as a rule, and so a relevant interpretation is the most natural. If this is the case, using an irrelevant system which can only simulate relevance in this kind of way would be akin to restricting our everyday arithmetic to one which could only simulate multiplication through a series of additions; theoretically workable but clumsy and unnecessary.

The other solution which will probably spring to mind is to rephrase the question as $Q = (A \rightarrow B) \land \neg B$, where we use ‘$-$’ to represent negation-as-failure (NAF). In words, ‘$B$ succeeds from the database with $A$ added, but not from the database itself’. Clearly this will be a sufficient condition for the relevant query $A \rightarrow B$ to succeed; if adding $A$ gives $B$ where $B$ was not previously inferrable, then the
required connection between the two must hold. However, it is not complete; for example, if our database consists of the elements \( A \rightarrow B, \ C \rightarrow B \) and \( C \), then \( Q \) will fail (because \(-B \) does), which cannot be what was intended. Proceeding along this path, one might think of the following: \( A \rightarrow B \) succeeds from \( T \) only if \( B \) succeeds from \( (T - B) + A \), that is, the result of first retracting \( B \) from \( T \) and then adding \( A \). Only the most intrepid would be willing to deal with the complexities of such a reading. We will restrict ourselves to two simple observations. First, there will always be more than one way of retracting enough information from \( T \) so that it no longer supports \( B \); \( T - B \) is not a uniquely defined object in the same way that \( T + A \) is (see Chapter 5 for more in this area). Secondly, consider the case where the database in the example also has the element \( A \). One way of cutting it down so that it no longer supports \( B \) is to remove both of \( A \rightarrow B \) and \( C \rightarrow B \); but clearly adding \( A \) to \( this \ T - B \) will not give something from which \( B \) can be inferred, and the query \( Q \) will again fail. Somehow or other we would have to ensure that any set not containing \( A \rightarrow B \) is not an acceptable choice for \( T - B \).

We can sum up the discussion of this section with a simple motto: 'Black-box' models are inappropriate for rule-like conditionals. By a ‘black-box’ we mean something like the collection of lightbulbs mentioned earlier, where the connections between them are hidden from us within a box. The bulbs have switches attached, which we can turn on, causing that bulb, and possibly others, to light. Turning on the switch corresponds to asserting something into the database; turning it off corresponds to a retraction of that element. However, since the connections within the box are such that a bulb \( A \) being on may cause another bulb \( B \) to be on also, just turning \( B \)’s switch off may not be enough to cause it to go out. Moreover, turning on another bulb \( C \) might cause the internals of the box to rearrange themselves in such a fashion that \( A \) is now connected to \( B \); this corresponds to adding an element \( A \rightarrow B \) in the database (note that implications are ‘first-class’ objects; there is no reason why they should not have the same status as ‘facts’). Under these circumstances there would seem to be no complete way of determining, for any given state of the box, whether or not there is an internal connection between \( A \) and \( B \), where the only operations we allow our-
selves are the turning on and off of switches (corresponding to asserting things into the database, in the style of NPROLOG, and also permitting ourselves to retract from it), and observing how this changes the state of the lights.

The black-box model is attractive because it conforms closely to the conditions of real world experiments where the goal is to form hypotheses about the hidden workings of some situation by observing the effects of manipulating those elements which are under our control. But this is not the correct model for a database, which contains the rules explicitly; they have already been found for us, somehow or other, and in determining whether or not \( A \) is connected to \( B \) we have rather the model of attempting to trace a pathway through this explicitly given connection-structure from one to the other. It is as if we had removed the cover from the box, exposing its innards. The model we use for the relevant conditional follows this pattern; to determine whether \( A \rightarrow B \) according to the given structure, we move back along the pathways leading to \( B \), in the hope of eventually reaching (i.e., ‘using’) \( A \).

### 1.3 A Planning Example

For this example we provide a more concrete instance of the abstract discussion of the last section. We imagine a geographic region divided, for whatever reasons, into zones. The atom ‘\text{state(zonel,polluted)}’ will be interpreted as asserting that zonel is polluted, for example. Take the following partial description \( T \) of the region:

\[
\begin{align*}
\text{r1} & \quad \text{state}(Z_1, \text{downwind-of}(Z_2)) \land \text{state}(Z_2, \text{factory}) \rightarrow \text{state}(Z_1, \text{polluted}) \\
\text{r2} & \quad \text{state}(Z, \text{toxic-dump}) \land \text{state}(Z, \text{populated}) \rightarrow \text{state}(Z, \text{danger-to-pop}) \\
\text{f1} & \quad \text{state}(\text{zonel}, \text{downwind-of(zonel2)}) \\
\text{f2} & \quad \text{state}(\text{zone4}, \text{toxic-dump}) \\
\text{f3} & \quad \text{state}(\text{zone4}, \text{populated})
\end{align*}
\]
We can imagine an environmental planner being interested in questions like 'What would happen if a factory were to be built in zone2?', 'Why is zone4 dangerous?' and so forth. To take the first question, we see how our natural intuitions about how to answer the question fit into the Ramsey Test mold: we ask the query 'state(zone2, factory)→state(X, Y)' ('What does having a factory in zone2 imply?') and in the simplest form of the Ramsey Test, this is equivalent to adding ‘state(zone2, factory)’ to \( T \), and asking the question 'state\((X, Y)\)'. This is the procedure which NPROLOG uses, for example. The answers we get are:

1. \( X = \text{zone2}, Y = \text{factory} \) (using just the antecedent of the query).
2. \( X = \text{zone1}, Y = \text{polluted} \) (using \( r1 \) and \( f1 \) and the added fact).
3. \( X = \text{zone4}, Y = \text{danger-to-pop} \) (using \( r2, f2 \) and \( f3 \)).
4. \( X = \text{zone1}, Y = \text{downwind-of(zone2)} \) (using \( f1 \)).
5. \( X = \text{zone4}, Y = \text{toxic-dump} \) (using \( f2 \)).
6. \( X = \text{zone4}, Y = \text{populated} \) (using \( f3 \)).

The last four answers illustrate why we might be particularly interested in a relevant conditional. They clearly have nothing to do with the question of a factory in zone2; in the intuitionistic logic which forms the basis of NPROLOG they result from the theoremhood of \( A\rightarrow(B\rightarrow A) \) (ie, the truth of \( A \) implies that anything implies \( A \)). On the other hand, the major distinguishing characteristic of relevant logics (for the positive case) is that they do not support \( A\rightarrow(B\rightarrow A) \), since for \( B \) to imply \( A \) in these systems there must be some real connection between \( B \) and \( A \), which is not the case in the last four answers above. In NPROLOG, an implicational goal \( A\rightarrow B \) is reduced by adding \( A \) to the database, and setting the new goal to \( B \). So \( A\rightarrow B \) will succeed whenever \( B \) does. Relevantly, however, \( A\rightarrow B \) succeeds only if \( B \) succeeds in some way which actually uses \( A \).

As to why this might be a useful property, consider the third answer. Clearly there is no way in which our planner could take this as an undesirable consequence of a factory in zone2, which was presumably one of the purposes of the question;
note that the same answer would be produced by an arbitrary query like ‘What happens if we build a mine in zone5?’. To labour the point, in CLOGPROG the following **bad-state** predicate makes sense:

\[
\text{f4 bad-state}(Z, \text{polluted}) \\
\text{f5 bad-state}(Z, \text{danger-to-pop}) \\
\text{r3 (state}(Z_1, S_1) \rightarrow \text{state}(Z_2, S_2)) \land \text{bad-state}(Z_2, S_2) \rightarrow \text{bad-state}(Z_1, S_1)
\]

(that is, a bad state is one which is directly undesirable, or one which contributes to a bad state), but it is inappropriate for an irrelevant system, where *everything* ‘causes’ a bad state, namely a dangerous zone4. We can imagine a wide range of general planning applications for more sophisticated forms of this simple definition; of course, we could also have the corresponding **good-state** predicate, defined in the same way, and it would be straightforward to attach some kind of explanation mechanism for expert systems applications.

In a relevant system, the success of a query \(A \rightarrow B\) amounts to a statement that a **rule** allowing us to conclude \(B\) from \(A\) can be inferred from the structure of the situation. This is not true for a non-relevant system; given the theorem scheme \(B \rightarrow (A \rightarrow B)\) we know that being able to infer \(B\) is sufficient to conclude \(A \rightarrow B\), but there is no natural sense in which we would want to say \(B\) is sufficient for \(A \rightarrow B\) to hold as a rule. A planner trying to determine the consequences of building a factory in zone2 will really want to know what rules are can be inferred dealing with such a circumstance, and this is the information which a relevant system provides.

To illustrate that the reading \((A \rightarrow B) \land \neg B\) (with \(\rightarrow\) taken irrelevantly) which we proposed and discarded in the last section is not appropriate here, consider adding the following to the database:

\[
\text{f4 state}(\text{zone1}, \text{downwind-of(zone3)}) \\
\text{f5 state}(\text{zone3}, \text{factory})
\]
Let us ask the question: ‘Would building a factory in zone2 lead to pollution in zone1?’ Given the new database, we can conclude that zone1 is already polluted, because of the factory in zone3, and so on the suggested reading the answer to the query would be ‘No’. We take it as obvious that this is not the correct answer to the query as stated. The correct interpretation of the suggested reading is: ‘Would building a factory in zone2 lead to pollution in an otherwise unpolluted zone1?’ which would no doubt be useful under certain circumstances but is not what we require here.

1.4 A Propaganda Example

An accidental example of the pernicious influences of the theorem scheme \( A \rightarrow (B \rightarrow A) \) is provided by [GR84], p335.

\[ d_1 \ (\text{friend}(X, Y) \land \text{criticises}(X, Y) \rightarrow \text{depressed}(Y)) \rightarrow \text{neurotic}(Y) \] (‘A person is neurotic if they are deeply depressed whenever one of their friends criticises them’ - Gabbay and Reyle’s reading, but see below.)

\[ d_2 \ \text{criticises}(\text{sally, mary}) \rightarrow \text{depressed}(\text{mary}) \] (‘Mary is depressed when Sally criticises her.’)

\[ d_3 \ \text{criticises}(\text{sally, X}) \] (‘Sally criticises everybody.’)

\[ d_4 \ (\text{friend}(X, Y) \rightarrow \text{neurotic}(Y)) \rightarrow \text{bad-friend}(X) \] (‘A person is a bad friend if there is someone such that if this person were her friend, that person would be neurotic.’)

With the query ‘\( \text{bad-friend}(X) \)’ (‘Who is a bad friend?’) Gabbay and Reyle produce a computation which gives the answer ‘Sally’; in fact in the example their handling of variables is faulty and the correct answer, given the intuitionistic reading of the conditional, is a surprising ‘Everybody’. The answer comes via two applications of our favourite enemy. First note that we cannot assert \( (A \rightarrow B) \rightarrow C \) (for any \( A, B, C \)) without in effect also asserting \( B \rightarrow C \); since \( B \rightarrow (A \rightarrow B) \) is a theorem, \( ((A \rightarrow B) \rightarrow C) \rightarrow (B \rightarrow C) \) must be also, by the Suffixing property which
holds for most logics. This together with simple quantifier moves (which hold relevantly as well as intuitionistically) tells us that ‘\( \text{neurotic}(Y) \to \text{bad-friend}(X) \)’ follows from the clause d4: translated, ‘If anybody is neurotic then everybody is a bad friend’, which while admirable as an expression of human oneness is hardly an expected consequence of the English version of d4. A similar derivation from d1 gives the equally unexpected ‘\( \text{depressed}(X) \to \text{neurotic}(X) \)’: ‘Anybody who is depressed is neurotic’. From d2 and d3 we can conclude that Mary is depressed; from the second of our derived clauses we conclude that Mary is neurotic, which by the first derived clause makes everybody a bad friend.

Note that in this answer we have not used any consideration of friendship. The initial goal of \( \text{bad-friend}(X) \) generates the goal \( \text{friend}(X,Y) \to \text{neurotic}(Y) \) (using d4) which in turn leads to the goal \( \text{neurotic}(Y) \) from the database with \( \text{friend}(X,Y) \) added. Readers can satisfy themselves that there is no way that this added element can ever get to be used; in fact the query fails relevantly. This may seem surprising given the English versions of the rules; the problem is in the formalization of d1 which should instead be

\[
\text{d1 } \text{friend}(X,Y) \land (\text{criticises}(X,Y) \to \text{depressed}(Y)) \to \text{neurotic}(Y)
\]

From Gabbay and Reyle’s reading we can derive the following, in (almost) any logic you care to name:

- \( (\text{friend}(X,Y) \to \text{depressed}(Y)) \to \text{neurotic}(Y) \) (‘Anybody who would be depressed were they to have a friend is neurotic.’)

- \( (\text{criticises}(X,Y) \to \text{depressed}(Y)) \to \text{neurotic}(Y) \) (‘Anybody who is depressed when criticised is neurotic.’)

Presumably these are not intended. With our reading both the relevant and intuitionistic systems produce the expected answer ‘Sally is a bad friend’. In the intuitionistic system we can no longer conclude from d1 that anybody who is depressed is neurotic, and so the fact Mary is neurotic no longer follows either. However the clause we derived from d4 still stands; were we to add an
additional piece of information to the database, 'John is neurotic' say, we would again conclude that everybody is a bad friend.

We like this example; it makes good propaganda for our belief that 'if $A$ then $B$' is generally taken as a rule; the clauses we derived from the original version of d1 and from d4, which Gabbay and Reyle surely did not intend, only follow because the natural reading fails in non-relevant systems.

1.5 Abduction, Reflection and Non-Regularity

Conditionals bring to the purely logical language of our systems some facility for representing the abductive process (see for example [Kow79, p241], [Esh88]); very generally, given some $A$, what $B$ will serve to explain it? In the circuit diagnosis example, we wanted to find the state-set($S$s) which would explain the input/output relation $\text{input}(I) \rightarrow \text{output}(O)$. In the planning example we could use $\text{state}(Z, S) \rightarrow \text{state}(\text{zone1}, \text{polluted})$ to return the states which would lead to pollution in zone1. It should be clear by now why a relevant reading of the conditional has an advantage here when the thing we want to explain is already in the database, or inferrable from it.

Generalizing on these ideas, we can see how conditionals might be used to make available (part of) the proof tree for some goal $A$ available to the program itself; in other words, to provide some primitive reflective facilities for the purely logical part of the language. For example, if the particular elements of the database we are interested in all have the form ‘$\text{state}(X)$’,

\[
\text{set-of}(X, \text{state}(X) \rightarrow A, L)
\]

will bind $L$ to a list of all the interesting elements which can be used in a proof of $A$. We could use this as the base for a more sophisticated treatment which delivers actual proof trees, rather than just the elements which appear in some proof tree. We can also extend it so that elements with different predicate symbols are included, and so on. The point is that we have the possibility of adding this kind of clause to an existing database in a modular fashion, whereas in a conventional
logic programming language we would have to tinker with the individual clauses in the database to include some usage record.

As a rather speculative example of how this kind of reflective facility might be useful, consider the query \( A \rightarrow A \). This is a theorem of relevant logic, so in the normal course of events we would expect it always to succeed for our logic programming system; in Ramsey Test terms, there is indeed a proof of \( A \) from \( A + T \) which uses \( A \), for any \( T \). Note that this proof will generally not use any element of \( T \); see, for example, the first answer produced in the planning example above. However, we do have the option of interpreting our database in such a way that it does not contain all theorems of the underlying logic; technically, it will correspond to a set of sentences which is closed under logical implication, but does not contain all the axioms. This is a perfectly coherent view of theoryhood; we investigate the logical issues in the next Chapter. Here we present an example where it may be advantageous to interpret the database in this 'non-regular' way.

Computationally, the difference is that \( A \rightarrow B \) follows from \( T \) only if some element of \( T \) (as well as \( A \)) is used in the proof of \( B \). \( A \rightarrow A \), therefore, will generally fail, and if it succeeds it will do so only in virtue of some information in the database allowing \( A \) to be deduced from itself. But this tells us that if \( A \rightarrow A \) succeeds from \( T \), then there is an infinite computation path for \( A \) from \( T \), specifically one which generates another call to \( A \). Thus we have the possibility of using a non-regular interpretation of the database as a loop-detection device. Note that this could not work for an irrelevant system, where \( A \rightarrow A \) will be inferrable from \( T \) whenever \( A \) is, even if we interpret the database non-regularly. We will postpone the details of the strategy until Sec. 4.7.

1.6 Relevance in Context

The lack of introductory texts for relevant logics means that a neophyte can quickly become lost in the tangle of intuitions surrounding the initially attractive dogma that \( A \) should only be said to imply \( B \) when there is some 'real connection' between the two, being left more or less to construct her own path through the
thicket. (In the next Chapter we present the results of our own efforts on these
lines.) Much of the relevant logic literature is highly technical and deals with
issues whose import is not readily apparent to 'outsiders'. The fact remains
that just about all logicians agree that most natural usages of 'if ... then' do
require such a connection; it would be hard to find an entry-level logic text which
does not admit at some early stage that the material implication to be discussed
therein fails to have any more than a tenuous relation to natural senses. Material
implication represents a lowest common denominator; any conditional will have
the property that when $A$ implies $B$, either $A$ is not true or $B$ is true. It also has
a simple and elegant logic, and in many situations serves as a good approximation
to a real implication, in the dark with the light behind it. On the other hand
it fails as an expression of a rule like conditional. The argument will be over
whether there exists a strand of natural usage coherent enough to formalise, and
if so whether relevant logics do it properly.

This brings us to a consideration of the objections which have been lodged
against relevant logics. On the issue of semantics, the most common formula­
tions, the Routley-Meyer relational semantics (see [SMPB82] for a summary)
and the Urquhart semi-lattice semantics ([Urq82]), are inaccessible to anyone not
already familiar with the issues of relevance; at least so we have found them.
We believe that the best approach, at least for explanatory purposes, is to be
found in the Dunn algebraic semantics ([Dun66], [MR72], [AB75, §28.2]), which
we recommend to anybody with strong semantic leanings. In this paper, how­
ever, most of our logical discussion is couched in proof theoretic terms, using
either Gentzen formulations or the methods of the next Chapter which are only
marginally semantic. Logical intuitions in the logic programming world seem to
be primarily model theoretic, and we expect that the lack of such a treatment
will count against the acceptance of our work. In defence, neither the relational
nor the semi-lattice semantics would provide a treatment in line with these in­
tuitions. It seems likely that an approach could be built on the Dunn algebras,
and this remains an interesting avenue for future work; as well, the formulation
we present in the next Chapter could probably be adapted along these lines. On
the other hand, we do not believe that our proof theoretic formulation is any less
intuitive than a model theoretic treatment would be, nor that such a treatment would provide a better system for explicating ‘meaning’ (which seems to be the claim made for model theory). The historical tradition within which we view our work stems from the Herbrand-Gentzen Theorem in its guise as Gentzen’s extended *Hauptsatz* ([Sza69, Ch. 3]).

A more important stumbling block to the wider acceptance of relevant logics is the inconsistency tolerating ‘DeMorgan’ negation which normally comes bundled with them. This point deserves discussion, if only to clear up some misconceptions about what relevant logic actually is. Besides rejecting $A \rightarrow (B \rightarrow A)$ as a valid theorem form, relevant logics as they are normally formulated are *paracconsistent*, which means that they also do away with $A \land \neg A \rightarrow B$ (‘A contradiction implies everything’ - ‘ex falso quodlibet’). As the well-worn Lewis argument ([LL32, p.250]) shows, we can only avoid ex falso if we are prepared to give up some principle which we might commonly regard as necessary:

1. Assume $A \land \neg A$
2. From 1, conclude $A$.
3. From 2, conclude $A \lor B$
4. From 3 and 1, conclude $\neg A \land (A \lor B)$.
5. From 4, conclude $B$ - if $A$ is false, and either $A$ or $B$ is true, then $B$ must be true, since $A$ cannot be both true and false.

The step which relevant logic rejects is 5: the argument is just that by assuming that $A$ is both true and false at step 1, we have moved out of the normal, comfortable world where contradictions never actually hold, to one where they do. The most we can do at 5 is conclude that *either* $B$ is true, *or* there is a contradiction on $A$, ie $(A \land \neg A) \lor B$. But the avoidance of $A \land (\neg A \lor B) \rightarrow B$ as a logical truth is unpalatable to many. On the other hand, the rejection of ex falso has interested some people in the computing world as a way of dealing sensibly with inconsistent information; presumably most large databases will have some kind of inconsistency, and it would be nice to have a logic which could take account of
that ([Bol85],[Bat86],[SW76],[MO86],[BS87] etc.). This interest is largely a result of Belnap’s proposals in [Bel77a],[Bel77b]. See Chapter 3 for further discussion.

An unfortunate consequence of this interest is that it has apparently led to a belief that paraconsistency is the characteristic feature of relevant logics, a belief which is just false (and not true). As Meyer has shown ([Mey79]), it is possible to take the positive fragment of $\mathbf{R}$ (the major relevant logic) and extend it with a ‘normal’ Boolean negation conservatively, that is in a way which does not affect the characteristics of the logic with regard to just those formulas which do not have any occurrences of negation. $A \rightarrow (B \rightarrow A)$ still fails to be a theorem of the new system, for example. It is unfortunate that the differences between logics are so often characterized in terms of their negations, when the backbone of any logic is its implication, everything else being more or less an add on extra. The belief seems to come about through the concentration on the first degree fragment of logics. A first degree formula has the form $A \rightarrow B$ where $A$ and $B$ contain no implications (in the standard vocabulary, their only connectives are $\land, \lor$ and $\neg$).

At this level, the positive fragments of the major logics coincide, so for example we can think of the positive part of normal PROLOG as classical, intuitionistic or relevant, according to prejudice; the differences only arise when we throw negation into the pot as well. But at least for relevant logics, the first degree fragment does not determine the properties of the full logic, and so the work which has been done on inconsistency tolerating first degree systems is does not subsume the our work here, which is concerned with the general degree. The choice of whether we extend the positive fragment of our system with DeMorgan negation, boolean negation, or some species of negation-as-failure, or any combination, is in principle left open: we still preserve the essential feature of relevant conditionals, namely that $A \rightarrow (B \rightarrow A)$ fails to be a logical truth. In fact, CLOGPROG does implement a form of DeMorgan negation (weaker than in the standard relevant logics - see Chapter 3), but there is no reason in principle why different forms of negation should not be able to co-exist within the general framework provided by the positive basis.

Another drawback to the use of relevant logics is their large computational com-
plexity. The major relevant logic, $R$, is undecidable even at the propositional level. The strongest fragment of it known to be decidable, $LR$, is nevertheless extremely expensive to compute with currently-known techniques (and perhaps essentially so - see [TMM88] for a discussion). For CLOGPROG we have found an unusual fragment of $LR$ (equally, so we believe, of $R$) which includes first-order features and which can be computed by a straightforward and efficient fragment to the standard SLD model, while retaining a large proportion of the expressive power of the full systems. See Chapter 4 for the details.

As we stated above, there is no real introductory text for relevant logic. The major texts in the field are [AB75] and [SMPB82]; perhaps the best starting point is [Dun86]. Some references to previous interest in full relevant systems in the AI community may be found in [MO86],[MS86], [TMM88], [SW76] and [AS88].
Chapter 2

A Formulation of Relevance

2.1 Rules, Entailments, Theories and Logics

Here we present formulations of the positive fragments of some relevant logics which are essentially the result of our own attempts to come to grips with the notion of a rule from basic intuitions. We hope that this deconstruction will serve as a more accessible introduction to relevance than is generally to be found in the literature, at least for people without highly-developed mathematical sensibilities. A drawback to the approach is that in attempting to reduce complex logical systems to simple pieces, we unavoidably end up with many pieces, which may in part defeat the purpose of the exercise. However the fine structure thus revealed allows distinctions to be drawn which may be obscured by more monolithic formulations, and we believe this to be a major advantage. The aim is to state the underlying intuitions as clearly as possible, and show that they are at least coherent. We leave the extremely contentious issue of negation until the next chapter, in part because the systems we construct there are not part of the standard relevant (or any other well-known) family.
CHAPTER 2. A FORMULATION OF RELEVANCE

2.1.1 Syntax of the Object Language

We define a language $L$ with connectives $\rightarrow$, $\neg$, $\vee$, $\land$, $\circ$, $\forall$, $\exists$ and $\neg$. All of these are binary connectives, except $\neg$, $\forall$ and $\exists$, which are unary. The connectives $\vee$, $\land$, $\forall$, $\exists$, $\rightarrow$ and $\neg$ are intended to take meanings which fall within their normal scope; the other connectives have interpretations which we will explain as we go along.

We help ourselves to a denumerable stock of n-ary predicates, function symbols and variables, constructing terms and atomic sentences in the normal way. We define well-formed formulas in the standard recursive fashion, allowing ourselves to eliminate parentheses with the understanding that all binary connectives associate to the left, with binding strengths (from weakest to strongest) following the order in which the connectives were stated above. We may also replace pairs of parentheses with dots; a single dot is to be taken as standing for a pair of parentheses bracketing the subformula which extends from immediately to the right of the dot to the immediate left of the first encountered right parentheses enclosing the subformula in which the dot occurs, or the end of the formula, whichever comes first. So $A \land B \lor (C \rightarrow D) \lor E \rightarrow F$ becomes $(A \land ((B \lor (C \rightarrow D)) \lor E) \rightarrow F)$ when all parentheses are restored in accordance with binding strength, associativity and dots.

An atom is defined to be either an atomic sentence, or (anticipating the next Chapter) $\neg p$ where $p$ is an atomic sentence. The symbols used for the various linguistic divisions are as follows:

Function Symbols $f, g, h, \ldots$

Variables $x, y, z, \ldots$

Terms in General $s, t, u, \ldots$

Predicate Signs $F, G, H, \ldots$

Atoms and Atomic Sentences $p, q, r, \ldots$

Wffs in General $A, B, C, \ldots$
We will let the context and the reader’s understanding resolve use-mention and other confusions.

Many of the systems to be discussed will be restricted to subsets of the full language, eg, the propositional part, or the positive part (without $\neg$), or the part with only the 'normal' connectives $\&$, $\lor$, $\to$ and $\neg$; and so on. In this chapter we will only be concerned with languages constructed from the positive part of $L$.

We define the degree of a sentence $A$ of $L$, denoted by $\text{deg}(A)$, as follows:

- $\text{deg}(p) = 0$ for all atoms $p$.
- $\text{deg}(\neg A) = \text{deg}((\forall x)A) = \text{deg}((\exists x)A) = \text{deg}(A)$.
- $\text{deg}(A \oplus B) = \max(\text{deg}(A), \text{deg}(B))$ where $\oplus$ is $\&$, $\lor$, $\lor$, or $\neg\rightarrow$.
- $\text{deg}(A \rightarrow B) = 1 + \max(\text{deg}(A), \text{deg}(B))$.

So the degree of a formula is a measure of the complexity of nested implication signs in it; for example, a zero-degree formula is one which contains no occurrences of $\rightarrow$, and a first-degree implication has the form $A \rightarrow B$ where $A$ and $B$ are zero-degree.

### 2.1.2 Entailment Systems

Let $T$ be some arbitrary set of sentences of $L$. If we take $T$ to be a partial description of some world, we can deduce from the elements of $T$ some more statements which hold of that world; if $T$ contains $A \& B$ we can deduce $A$, if it contains $A$ we can deduce $A \lor B$, and so on. This is because we take it that the logical meaning of $A$ is contained in that of $A \& B$, and similarly $A \lor B$ is contained in $A$. The backbone of our formulation is this meaning containment relation, which we will express formally by the symbolism ‘$A \Rightarrow B$’. We develop formal systems in which these ‘$\Rightarrow$’ statements are the objects; these systems serve to define the meanings of the logical connectives, for we will take it that the logical meaning of an $L$ sentence $A$, according to one of these systems, is just the set of $B$ for which $A \Rightarrow B$ is derivable in that system. Thus our systems will always contain
as axioms $A \rightarrow A \lor B$ and $A \land B \rightarrow A$, for example. To some extent, then, these systems provide a semantics, in that meaning is presumably a semantic notion; but the systems themselves are just formal systems in the sense of [Cur77], and the operations we perform with them and on them are characteristically proof theoretic.

We will call the systems *entailment systems*, and we will read $A \Rightarrow B$ as an entailment, sanctioned by the meaning relationship. An entailment system $\mathcal{E}$ is specified by giving some initial objects ('axioms'), and some rules for generating new objects from old. The property $\vdash_{\mathcal{E}} A \Rightarrow B$ ('$A \Rightarrow B$ is derivable in $\mathcal{E}$') is defined in the normal inductive way:

- $X_0$ is the set of axioms of $\mathcal{E}$.
- $X_{n+1}$ is the union of $X_n$ and the set of all $A \Rightarrow B$ which follow from elements of $X_n$ by application of some $\mathcal{E}$ derivation rule.
- $\vdash_{\mathcal{E}} A \Rightarrow B$ iff $A \Rightarrow B \in X_n$ for some $n$.

We will use the short hand notation $\vdash_{\mathcal{E}} A \leftrightarrow B$ to indicate that both $A \Rightarrow B$ and $B \Rightarrow A$ are derivable.

An important feature of these entailments is that they are to be taken as totally non-elliptic: $A \Rightarrow B$ is intended to state that the logical meaning of $A$, by itself, without the help of any other statements, is totally sufficient to deliver $B$. Even if we are operating in the context of a theory of truth which says that some sentences of $L$ always hold, we cannot assume any of those things in determining whether $A \Rightarrow B$ obtains. For example, say that we take some sentence $B$ to be a 'logical truth'. Then even though, as it happens, our systems will always support $A \land B \Rightarrow B$, we are not allowed to conclude therefore that $A \Rightarrow B$, for arbitrary $A$.

Clearly, then, the $\Rightarrow$ relation does not correspond to classical accounts of logical consequence, in which $A \Rightarrow B$ holds just in case it is not possible for $A$ to be true and $B$ not to be true, for on that reading, $A \Rightarrow B$ must hold whenever it is impossible for $B$ to be not to be true, regardless of $A$. We will not recapitulate the arguments against the classical position here; good starting points may be
found in [AB75] and [SMPB82]. We will assume that readers acknowledge at least 
the prima facie plausibility of a view on which, for example, the logical meaning 
of 'Sheep are green' cannot be said to include 'My name is Andrew implies my 
name is Andrew'; more precisely, a view on which it is not the case that logical 
truths have 'no content'.

2.1.3 Theories

Every entailment system \( \mathcal{E} \) induces a class of subsets of \( L \) closed under \( \mathcal{E} \)-
entailment:

\[
\text{If } A \in T \text{ and } \vdash \mathcal{E} A \Rightarrow B \text{ then } B \in T.
\]

That is, those sets which respect the \( \mathcal{E} \) meaning-containment relation. We will 
call these the \( \mathcal{E} \)-theories. Note that there is no requirement that \( T \) must contain 
anything; \( \emptyset \) will always be a \( \mathcal{E} \)-theory for all of our entailment systems. In fact 
we will only be interested in sets which are also closed under:

Adjunction If \( A, B \in T \) then \( A \& B \in T \).

In general, however, we will not require the dual property of primeness; our 
theories need not be closed under:

\[
\text{If } A \lor B \in T \text{ then } A \in T \text{ or } B \in T
\]

We take it that this asymmetry is not unnatural; we expect that when a theory 
supports both \( A \) and \( B \) it will support the conjunction of the two, but we will 
not be surprised if it fails to support each of \( A \) and \( B \) even when it tells us that 
their disjunction holds. From now on we will assume that all theories are closed 
under Adjunction.

2.1.4 Logics

The apparatus sketched thus far does not serve to designate a non-empty sub-
class of necessary truths from the set of sentences of \( L \). A necessary truth is
standardly taken to be a sentence \(^1\) which cannot, under any circumstances, fail
to hold. Alternatively, one which holds just in virtue of the meaning assigned
to its connectives, regardless of the meaning of its constituent atomic parts. If
we gloss the first characterization as ‘is an element of every \(\mathcal{E}\)-theory’ (for some
entailment system \(\mathcal{E}\)) then we have no necessary truths, since the empty set is
always a \(\mathcal{E}\)-theory. On the other hand, the only objects discussed so far which
can be said to hold in virtue of their form alone are the statements provable in
an entailment system, which are not part of the language \(L\).

We have two choices. The first is to leave things as they are, abandoning any
concept of necessary truth for sentences of \(L\), while retaining the notion for
elements of the entailment systems; in other words, only recognize statements
concerning meaning containment as having the character of necessity. The other
is to use some additional criteria to pick out a subclass \(X\) of necessary truths
from \(L\) and assign a special status to the \(\mathcal{E}\)-theories which contain \(X\). We will
call these the regular theories (with respect to the theory of truth embodied in
\(X\)).

Concentrating on the second alternative for the moment, the obvious question
arises as to the status of the non-regular theories, those which do not contain
all the necessary truths. Most importantly, they can serve to give a sensible
characterization of the meaning of a formula; the set \(\{B: \vdash_{\mathcal{E}} A \Rightarrow B\}\) will always
be the smallest \(\mathcal{E}\)-theory which contains \(A\). If instead we were to adopt the more
standard picture, and say that the meaning of \(A\) is the smallest regular \(\mathcal{E}\)-theory
containing \(A\) then we would have to include all necessary truths, and this seems
to us to be just wrong. As we noted above, we do not believe that the meaning
of ‘Sheep are green’ includes that of ‘My name is Andrew implies my name is
Andrew’ (where for the sake of the argument we take \(A \rightarrow A\) to be a necessary
truth). On a related note, real scientific theories do not seem to be construed
in a regular way; Newtonian physics would not normally be said to include the
proposition ‘If \(X\) is a particle then \(X\) is a particle’.

The notion of necessary truth which we will adopt here rests on the view that

\(^1\) Or proposition, depending on one’s philosophical bias; we will ignore these issues here.
the proper business of logic is the determination of what follows (logically) from what. For each \( \mathcal{E} \) we define the class of regular \( \mathcal{E} \)-theories to be those which reify \( \mathcal{E} \)-entailment in the sense that \( A \rightarrow B \) is an element of the theory whenever \( \vdash \mathcal{E} A \Rightarrow B \). For each of our systems, there is a smallest regular \( \mathcal{E} \)-theory; we think of this as the set of logically valid (necessarily true) sentences according to the notion of consequence embodied by \( \mathcal{E} \), and call it the \( \mathcal{E} \)-logic. As [Mat72, p7] puts it, 'an argument ... is sound if and only if the corresponding conditional is necessarily true'. In fact we depart somewhat from this statement of the orthodox position; while it is the case that the converse of the reification holds for our weaker systems, so that \( \vdash \mathcal{E} A \Rightarrow B \) whenever \( A \rightarrow B \) is an element of the \( \mathcal{E} \)-logic, stronger systems read the \( \mathcal{L} \) connective \( \rightarrow \) in an enthymematic sense, and the relationship will be: if \( A \rightarrow B \) is a logical truth then \( A \) perhaps together with some other logical truths entails \( B \). In contrast, we reiterate that entailment is always non-enthymematic in our systems; just because \( B \) is contained in the meaning of \( A \) and some logical truth acting together, there would seem to be no reason for assuming that \( B \) is contained in \( A \), for to do so seems tantamount to accepting the proposition that necessary truths have no content.

For most natural readings of \( \rightarrow \), taking \( A \rightarrow B \) to be true whenever the logical meaning of \( B \) is contained in that of \( A \) is intuitively plausible. However, note that it would presumably fail if we were to read \( A \rightarrow B \) as '\( A \) is a sufficient cause for \( B \)', for example, all of our systems will tell us that \( A \Rightarrow A \), but it seems incorrect to assume that \( A \) causes \( A \). For such a reading, the non-regular \( \mathcal{E} \)-theories are more appropriate. There is also a strong historical tradition, beginning with Aristotle, for a non-peripatetic reading of implication which rejects \( A \rightarrow A \) as a logical truth, the idea being that true implication should always lead to something different to the antecedent. Our own modest opinion is that the most coherent reading of this position is that true implication should always be productive, in the sense that the meaning of the conclusion is not contained in the antecedent, so on this view none of the elements of the '\( \mathcal{E} \)-logic' are logical truths; again, the appropriate objects are the non-regular theories.

For our logic programming systems, we will generally take the database as repre-
senting a regular theory, so that queries like \( A \rightarrow A \) will always succeed. However, the considerations of the last paragraph indicate that there are circumstances under which this is not appropriate. There are also pragmatic considerations (which may be referred to the notion of a ‘productive implication’) for taking the opposite view in some cases; we hinted at some of these in the last chapter, with the loop detection example (to be developed more fully later), and with the first ‘trivial’ answer to the planning example.

2.1.5 Summary

All of our logical apparatus is generated by the concept of meaning embodied in the entailment systems. Theories are sets of sentences closed under this meaning relation, and logics are special theories which reify it. This should be compared with Meyer’s (sadly unpublished) [Mey74], our main inspiration. There, the logic is taken as the primary object; for a given logic \( \mathcal{L} \), we define \( \mathcal{L} \)-theories as sets of sentences closed under \( \mathcal{L} \) logical implication, and regular \( \mathcal{L} \)-theories to be those which also contain all theorems of \( \mathcal{L} \). Logical implication in \( \mathcal{L} \) and the concept of entailment appropriate to \( \mathcal{L} \) are taken as equivalent; our major departure is the breaking of this correspondence. As we noted, our stronger systems will read implication enthymematically, whereas the underlying entailment system is always non-enthymematic. Our method, we believe, has the advantage of showing how the major relevant logic \( \mathcal{R} \), which in its normal formulations is often quite inaccessible for the novice, can be constructed using simple intuitions about meaning containment. In other words, we believe our method has an advantage when it comes to explaining why particular formulas are taken as necessary truths in relevant logics.

2.2 Extensional Conjunction and Disjunction

We start with the properties for familiar extensional conjunction and disjunction. Think of the logical meaning of a sentence of \( \mathcal{L} \) as a beaker containing all the
B whose meaning is included in that of A; label this beaker ‘A’. Extensional conjunction corresponds to a *mixing* operation; the beaker A & B contains the contents of A and B mixed together. This gives straightforwardly the following principles for ‘&’:

- **Axioms.**
  
  - **Idempotence**
    
    \[ I & A \Rightarrow A \land A \]
  
  - **Weakening**
    
    \[ W & A \land B \Rightarrow A \]
  
  - **Commutativity**
    
    \[ C \land A \land B \Rightarrow B \land A \]
  
  - **Associativity**
    
    \[ A \land A \land B \land C \Rightarrow A \land (B \land C) \]

- **Rule.**
  
  - **Monotonicity**
    
    \[ M \land A \Rightarrow B \land A \land C \Rightarrow B \land C \]

For example, \( I \land \) says that A contains whatever you get when you mix it with itself. We may as well introduce the rule which says that containment is transitive here, since it is so basic:

**Entailment Transitivity**

\[ T \Rightarrow A \Rightarrow B, B \Rightarrow C \Rightarrow A \Rightarrow C \]

Using just the principles given so far, it is simple to derive the converses of \( I \land \), \( C \land \) and \( A \land \) (though not, of course, that of \( W \land \)).

In terms of our beaker model, extensional ‘or’ corresponds to a *separation* operation. The beaker labelled ‘A ∨ B’ will look like one containing oil and water
(or a Gin Tropical): distinct layers with the contents of A and the contents of B separated from each other. We have to complicate our idea of what the contents of such a beaker are: we take these to be neither the contents of A nor those of B but the common contents of A and B (that is, of each layer). Clearly A will contain the contents of A \lor B, for any B, since whatever is common to A and B must be contained in A. Similar considerations serve to validate the following natural duals to the conjunctive principles:

- **Axioms.**
  
  **Idempotence**
  \[ \lor A \land A \Rightarrow A \]
  
  **Weakening**
  \[ \lor A \Rightarrow A \land B \]
  
  **Commutativity**
  \[ \lor A \land B \Rightarrow B \land A \]
  
  **Associativity**
  \[ \lor A \land B \land C \Rightarrow A \land (B \land C) \]
  
- **Rule.**
  
  **Monotonicity**
  \[ \lor A \Rightarrow B \Rightarrow A \land C \Rightarrow B \land C \]

The final zero-degree extensional principle tells us what that when we mix the contents of A with the contents of B \lor C, the result is a separation between the mixture of A and B and the mixture of A and C:

**Distribution**

\[ \lor A \land (B \lor C) \Rightarrow A \land B \lor A \land C \]

The converses of \lor, \lor, \lor and \lor \lor are derivable using just the principles listed so far.
2.3 Intensional Conjunction and Disjunction

In this section we give principles for ‘fusion’ and ‘fission’, intensional forms of conjunction and disjunction, which we represent by the connectives ‘\( \odot \)’ and ‘\( \oplus \)’ respectively. Where ‘\( A \& B \)’ was the simple mixture of the contents of \( A \) and \( B \), ‘\( A \odot B \)’ is what we get when we make the two react together; the ‘molecules’ in \( A \) form strong bonds with those of \( B \), yielding reaction products which may have none of the properties of their constituents, in the same way that H\(_2\)O has none of the properties of hydrogen and water. Clearly, then, we do not want the analogue of the Weakening principle \( W \& \) for fusion. We assume that our substances are such that everything can form these kind of bonds with everything else (including itself), and that the process of bonding is ‘spontaneous’, so that it will happen whenever we mix things together. This gives us the analogue of \( I \& \); we will also want to be able to derive the ordering principle \( A \& B \Rightarrow A \odot B \) (but not its converse). We take it that bonding is commutative, and the model clearly requires monotonicity. Associativity will follow under the assumption that the steps in any ‘reaction path’ can be performed in any order. Our principles for ‘\( \odot \)’, then, are just the direct translation of the extensional conjunctive principles, except that we drop Weakening. Explicitly:

- **Axioms.**
  
  **Idempotence**
  \[
  I_\odot \quad A \Rightarrow A \odot A
  \]

  **Commutativity**
  \[
  C_\odot \quad A \odot B \Rightarrow B \odot A
  \]

  **Associativity**
  \[
  A_\odot \quad A \odot B \odot C \Rightarrow A \odot (B \odot C)
  \]

- **Rule.**

  **Monotonicity**
  \[
  M_\odot \quad A \Rightarrow B \vdash A \odot C \Rightarrow B \odot C
  \]
We can derive the ordering principle mentioned above, and the converses of $C_0$ and $A_0$, but not that of $I_0$.

Intensional 'or' is perhaps more familiar in natural usage than its conjunctive counterpart; it is the type of disjunction which requires that its disjuncts be real alternatives, in a sense which would not support $A \Rightarrow A + B$ for arbitrary $B$. The first modern discussion of such a connective that we know of is in Lewis' [Lew14], but $A + B$ is taken as equivalent to $\neg A \Rightarrow B$ which is not generally the case for our systems; see Sec. 3.7. A relevant perspective on Lewis' intensional disjunction may be found in [SMPB82, p?]. Parry's analytic implication ([Par31]) discards $A \Rightarrow A \lor B$ (while retaining $A \& B \Rightarrow A$); see [SMPB82, Sec. 2.5] for a critique.

In terms of our chemical model, we think of 'A + B' as a beaker which not only contains a separation into an 'A' layer and a 'B' layer, but one where this separation is moreover forced by 'repulsive' bonds between the molecules of A and B. More clearly, the molecules of A and B form 'disjunctive compounds' in which the bonds serve to keep the constituents as far apart as possible. The contents of A will not in general contain the contents of A + B, because they will not include these disjunctive compounds. Notice that the molecules of A and B retain their individual characters in A + B. In a sense, the only 'new' elements in A + B are the disjunctive compounds, in contrast to A o B, where the attractive bonding between molecules is such that they lose their separate identity and may fuse together to produce something new. We do not assume that the formation of disjunctive compounds is spontaneous; perhaps the best picture is of substances for which the conjunctive compound represents a 'lower energy state', so that things will naturally tend to combine, whereas to get the disjunctive compound we need to add energy to the system.

On this model, we would expect to have Monotonicity, Commutativity and the intensional analogue of the Idempotence principle for 'V', as well as the ordering principle $A + B \Rightarrow A \lor B$. We will also assume that the disjunctive bonding process is not order-dependent, giving Associativity as well. Thus our principles for fission are just the direct translations of those for 'V', except that we drop Weakening:
• Axioms.

Idempotence

\[ I+ A+A \Rightarrow A \]

Commutativity

\[ C+ A+B \Rightarrow B+A \]

Associativity

\[ A+ A+B+C \Rightarrow A+(B+C) \]

• Rule

Monotonicity

\[ M+ A \Rightarrow B \Rightarrow A+C \Rightarrow B+C \]

We can derive the ordering principle mentioned above, and the converses of \( C+ \) and \( A+ \) but not, of course, that of \( I+. \)

This does not complete the story for ‘and’ and ‘or’; we still have to settle the question of how the connectives interact. To take ‘\&’ and ‘\lor’ first: if we react the contents of \( A \) with the contents of a beaker split into a \( B \) and a \( C \) layer, presumably the result will be split into a layer with the reaction of \( A \) and \( B \), and a layer with the reaction of \( A \) and \( C \):

\[ D \& V A \&(B \lor C) \Rightarrow A \& B \lor A \& C \]

The distribution of ‘\&’ over ‘+’ is not given by the direct translation of this principle. Remember that the contents of \( B+C \) are disjunctive molecules; when a molecule from \( A \) reacts with one of these, it will bind itself either to the ‘\( B \)’ end or the ‘\( C \)’ end; this is the natural consequence of our stipulation that the constituents of a disjunctive molecule retain their separate characters. The principle we want is:

\[ D \&+ A \&(B+C) \Rightarrow A \& B+C \]
CHAPTER 2. A FORMULATION OF RELEVANCE

Note that we can derive the weaker distribution form \( Ao(B+C) \Rightarrow AoB + AoC \), and also of course \( Ao(B + C) \Rightarrow B + AoC \). That we do not want \( \& \) to distribute over \( + \) should be readily apparent. If we were to have \( A\&(B+C) \Rightarrow A\&B + A\&C \) this would tell us that mixing the contents of \( A \) with the disjunctive compounds of elements in \( B \) and \( C \) would give us disjunctive compounds of elements from \( A \) and from \( C \), for example, which we have no reason to believe. However, the following ‘dual’ to \( D\&V \) seems to be justified (remembering that constituents retain their characteristics in disjunctive molecules):

**Associativity**

\[ A \& (A+B) \& (A+C) \Rightarrow A+B \& C \]

The analogous ‘\( A\&V \)’ and ‘\( A\&o \)’ principles are derivable; we leave it readers to convince themselves that we would not want an ‘\( A\&o \)’ principle. We conclude by noting the derivability of two principles giving the interaction of the conjunctive and disjunctive forms respectively (but not their converses):

- \( Ao(B\&C) \Rightarrow AoB \& AoC \)
- \( A+B \lor A+C \Rightarrow A+(B \lor C) \)

### 2.4 Implication

As discussed in the last chapter, part of the meaning of ‘\( A \rightarrow B \)’ in our systems is to be given given by the notion of a rule connecting \( A \) and \( B \). Our broad reading is: ‘\( A \) is sufficient for \( B \)’ in the sense that ‘If \( A \) were true then \( B \) would therefore be true’. The use of the subjunctive here is intended to distance the statement from any consideration of whether \( A \) and \( B \) are actually true or not, and to focus instead on the connection between them. Specifically, although we can reasonably conclude from the statement (if it holds) that either \( A \) is not true, or \( B \) is true, the converse will fail. From the fact that \( A \) is not true we cannot conclude that its truth would be sufficient for the truth of \( B \), and from the truth of \( B \) we cannot conclude that \( A \) being true forces the truth of \( B \), in the way...
required. Our implication is an intensional connective (as against an extensional one, like & or V; this just means that the status of a sentence like \( A \rightarrow B \) is not determined by the status of its parts). Fine tuning of this reading will centre around consideration of the sense of ‘therefore’.

2.4.1 Basic Principles

Our chemical analogy becomes even more strained here than it was for conjunction and disjunction; however we can perhaps wring a few more drops of motivational juice from it. We will say that the beaker labelled ‘\( A \rightarrow B \)’ contains substances which act as producers of the contents of B in the presence of the contents of A. For each molecule b in the contents of B, \( A \rightarrow B \) contains a ‘conditional molecule’ consisting of a kind of ‘deactivated’ instance of b with a ‘recognizer’ for the contents of of A attached to it. We think of the recognizer as being a kind of mirror image of A, into which the contents of A can fit in a key-and-lock manner, resulting in the activation of the b molecule. An immediate consequence of this model is the first Monotonicity principle:

\[
M \rightarrow_1 B \Rightarrow C \vdash A \rightarrow B \Rightarrow A \rightarrow C
\]

The second Monotonicity principle rests on intuitions about the mirror-image quality of recognizers:

\[
M \rightarrow_2 A \Rightarrow B \vdash B \Rightarrow C \Rightarrow A \rightarrow C
\]

If B is contained in A, the recognizers for A are contained in those for B.

Principles giving the interaction of implication with conjunction and disjunction follow if we stipulate that conditional molecules with the same recognizers react in the normal way, and that those with the same consequents react dually:

**Strengthening**

\[
S \& (A \rightarrow B) \& (A \rightarrow C) \Rightarrow A \rightarrow B \& C
\]
CHAPTER 2. A FORMULATION OF RELEVANCE

$S \lor (\neg C) \land (B \rightarrow C) \Rightarrow A \lor B \rightarrow C$

In logical terms, the Strengthening principles seem fundamental: if $A$ is sufficient for both $B$ and $C$, it is presumably sufficient for $B \& C$ also. Dually, if both $A$ and $B$ are sufficient for $C$, $A \lor B$ should be as well. Similarly for the Monotonicity rules; if the meaning of $C$ is contained in that of $B$, and $A$ is sufficient for $B$, then $A$ must be sufficient for $C$, and if $B$ is sufficient for $C$ and the meaning of $B$ is contained in the meaning of $A$, then $A$ will be sufficient for $C$.

Note that the chemical model fits better with our notion of a ‘productive implication’ in apparently not giving any preferred status to producers of the form $A \rightarrow B$ for the case where $B$ is contained in $A$; for there is no particular reason to think that this containment is mediated by conditional molecules.

We can derive the converses of the Strengthening principles using Monotonicity and Weakening, and we also have the following:

- $(A \rightarrow B) \& (C \rightarrow D) \Rightarrow A \& C \rightarrow B \& D$
- $(A \rightarrow B) \& (C \rightarrow D) \Rightarrow A \lor C \rightarrow B \lor D$
- $(A \rightarrow B) \lor (C \rightarrow D) \Rightarrow A \& C \rightarrow B \lor D$

It should go without saying that we do not want the archetypically irrelevant principle

$A \Rightarrow B \rightarrow A$

2.4.2 Implication with Fusion and Fission

The intensional forms of the Strengthening principles follow on the same model as we used for their extensional counterparts:

Strengthening

$So (A \rightarrow B) \& (A \rightarrow C) \Rightarrow A \rightarrow B \& C$
S+ \ (A \rightarrow C) \circ (B \rightarrow C) \Rightarrow A + B \rightarrow C

Alternatively, the following seem plausible:

- \ (A \rightarrow B) \circ (C \rightarrow D) \Rightarrow A \circ C \rightarrow B \circ D
- \ (A \rightarrow B) \circ (C \rightarrow D) \Rightarrow A + C \rightarrow B + D
- \ (A \rightarrow B) + (C \rightarrow D) \Rightarrow A \circ C \rightarrow B + D

by analogy with the corresponding extensional forms; So and S+ are derivable from these, but not vice versa. (Note also that the converses of the Strengthening principles are not derivable; recall that the converses of the extensional Strengthening principles required Weakening, which does not hold for the intensional connectives). In the absence of firm intuitions to discriminate between these alternatives, we will stick with our chemical model which only lets conditional molecules react when they have the same recognizers or the same consequents, thus picking out So and S+ as the preferred principles.

2.4.3 Elimination Principles

Conspicuous by their absence so far are the following principles which we take to be straightforward consequences of our reading of \( A \rightarrow B \):

Transitivity

\( T \circ (A \rightarrow B) \circ (B \rightarrow C) \Rightarrow A \rightarrow C \)

Detachment

\( D \circ A \circ (A \rightarrow B) \Rightarrow B \)

In terms of the chemical model, \( T \circ \) says that when the contents of \( A \rightarrow B \) react with those of \( B \rightarrow C \), the consequents of the the conditional molecules in \( A \rightarrow B \) 'lock into' the recognizers of the molecules in \( B \rightarrow C \), in such a way that the
recognizers for B are effectively replaced by those for A. More straightforwardly, Do just states what the action of a conditional molecule is.

To and Do share a feature which distinguish them from all the principles presented previously - namely that something appearing on the left in an intensional context disappears on the right. We will call any principle with this feature an elimination principle. (Note that \( W \& - A\&B \Rightarrow A \) is not an elimination principle on this definition, since the B on the left does not appear in an intensional context). Something on the left is 'used up' in the reaction which produces the new elements on the right. The reason for not including Transitivity and Detachment among the basic principles is related to this characteristic; the behaviour of systems with these will differ in important ways from those without, when we come to add negation, and it will be instructive to highlight elimination as the root of this difference.

Using the derived ordering principle \( A\&B \Rightarrow A\&B \) we also have the extensional conjunctive forms of Transitivity and Detachment:

\[ T \& (A \rightarrow B) \& (B \rightarrow C) \Rightarrow A \rightarrow C \]

\[ D \& A \& (A \rightarrow B) \Rightarrow B \]

The remaining elimination principle, Enthmeme, will be important enough in the development to deserve a discussion of its own.

2.4.4 Enthymemes

The principles adduced so far give a strong sense to the 'therefore' in our interpretation of \( A \rightarrow B \). We want now to consider weakening this sense a little. To illustrate, take a situation where 'Were A and B true then C would therefore be true' holds, and A does also. There are strong intuitions to support an elliptic sense for implication in which we would want to say that 'Were B true then C would therefore be true' holds here also, weakening 'therefore' to a sense in which the truth of B has just some force in making C true (in collaboration with the truth of A), rather than the total sufficiency assumed up till now. For example,
consider an electrical circuit in which two switches, $A$ and $B$, both need to be turned on for a light bulb $C$ to shine. If $A$ is already switched on, there is a natural sense in which we would want to say that 'Were $B$ switched on, $C$ would therefore shine' holds of this situation. This kind of implication, for which we want to say that $B$ implies $C$ in the case where $B$ together with some true $A$ really implies $C$, is traditionally called an enthymeme.

However a simple-minded approach to enthymemes leads quickly to irrelevance. Taking as our enthymeme principle the following:

$$A \land (A \land B \rightarrow C) \Rightarrow B \rightarrow C$$

we can see that $B \rightarrow C$ will be an element of any theory which contains both $A$ and $A \rightarrow C$, since $A \rightarrow C \Rightarrow A \land B \rightarrow C$ is derivable using $M \rightarrow 2$ and $W &$. Worse, any regular theory which contains $A$ will also contain $B \rightarrow A$ for every $B$, since such theories will always contain $A \land B \rightarrow A$ (the reified form of $W &$). All sense of 'therefore' has been lost. What went wrong? The answer lies in the use of $W &$ in the step from $A \rightarrow C$ to $A \land B \rightarrow C$; looking back to our lightbulb example, we implicitly assumed that both $A$ and $B$ were somehow connected to $C$, even if not in a way which made the setting of either a totally sufficient condition for $C$ to light, whereas here we have introduced a $B$ which has nothing to do with $C$.

The intensional conjunction we defined in the last section gives us a means of expressing the fact that $C$ depends on the joint action of $A$ and $B$; the connection in the example is best represented as '$A \circ B \rightarrow C$', because $C$ being lit results from $A$ and $B$ working together. The distinguishing characteristic of fusion is that it does not support Weakening; thus we would not be able to conclude $A \circ B \rightarrow C$ from $A \rightarrow C$ for an arbitrary $B$, and the following form of enthymeme principle will not result in irrelevance:

**Enthymeme**

$E \ A \circ (A \circ B \rightarrow C) \Rightarrow B \rightarrow C$

Note that $A \land (A \circ B \Rightarrow C) \Rightarrow B \rightarrow C$ is derivable from $E$, given the ordering principle $A \land B \Rightarrow B \circ C$, but of course $A \land (A \land B \Rightarrow C) \Rightarrow B \rightarrow C$ does not follow. Like
Transitivity and Detachment, Enthymeme is an elimination principle, since the ‘A’ on the left disappears on the right; the conclusion of the rule is to be seen as a ‘reaction product’ of the conditions. In the chemical model, we can see a reaction in which A ‘locks into’ part of the A o B recognizer, converting it into a recognizer for the remaining part, that is, for B.

2.4.5 Enthymeme vs Residuation

We want to compare the Enthymeme principle with one which may be seen as a competitor:

Residuation

1. **R** A o B =⇒ C ⊥ B =⇒ A =⇒ C

Were we to adopt Residuation rather than Enthymeme, our development would end up looking much more like the standard formulations of relevant logics (at least for strong systems), in fact almost (but not quite) isomorphic to Dunn’s algebraic semantics (see Sec. 2.11). Residuation also has a clear connection with the Ramsey Test mentioned in Chapter 1, and we do not dispute that the formal properties which it gives rise to have many advantageous features. Note that Enthymeme is derivable given Residuation, Detachment, Monotonicity and Commutativity principles for ‘o’: the last three of these give Bo Ao (AoB → C) =⇒ C, and Ao (AoB → C) =⇒ B =⇒ C follows by Residuation. However, Residuation does not follow from any of the other principles given so far, so it is essentially stronger than Enthymeme. (However, we will see in Sec. 2.11 that the regular theories, and the logics generated by Enthymeme and Residuation systems generally coincide.)

To illustrate, consider what happens when we residuate AoB =⇒ AoB (which is of course derivable, using I& and W&) to produce B =⇒ A =⇒ AoB. This says that the logical meaning of B contains the information that A is sufficient for the ‘reaction products’ of A and B. Now there is of course a clear sense in which this is highly plausible, but we would claim it is an elliptic sense. Bo (AoB → AoB) =⇒ A =⇒ AoB is derivable using E and simple moves, and B =⇒ A =⇒ AoB would follow if we
were to allow ourselves to ‘hide’ the assumption of $A \circ B \rightarrow A \circ B$ on the left; this suppression is of course not allowed by our non-elliptic view of entailment. In terms of the chemical model, the issue is whether we want to take the contents of $B$ to include, for every $A$, a conditional molecule with a recognizer for $A$ for every molecule in $A \circ B$. Again, this is not implausible; $A$ and $B$ do in fact react (everything does) and we could, if we liked, see the reaction as mediated by conditional molecules of this kind. On the other hand, such a view seems unnecessary; for example, the reaction of sodium with chlorine to produce salt does not depend on some substance \textit{contained in sodium} to mediate the reaction (unless we know even less about chemistry than we thought we did).

Residuation is an extremely strong principle. \textit{All} of the implicational principles we have so far adduced can be derived using just zero-degree principles, $Do$ and Residuation (see Sec. 2.11). Formally, this is a very attractive property; on the other hand, one of the reasons we prefer Enthymeme is that it provides what seems to be a natural way of characterizing the difference between the strong relevant logic $R_+$ (the positive fragment of $R$) and weaker systems. Specifically, we will show that $R_+$ can be seen as an \textit{enthymematic extension} to the weak relevant logic $DA_+$. In other words, the difference is that the implication of $R_+$ is enthymematic, whereas that of $DA_+$ is not. Now it would also be possible to characterize $R_+$ as a \textit{residuating extension} to $DA_+$, since adding Residuation to the principles which generate the logic $DA_+$ would certainly give an entailment system which generates $R_+$. But given the strength of Residuation, sufficient with $Do$ and zero-degree principles to deliver all the implicational principles of $DA_+$, such an approach could not really be seen as \textit{building up} to the stronger system. The Enthymeme approach, on the other hand, does give us such a picture. We believe that such a construction gives a more accessible path to $R_+$ than any others we have encountered; $DA_+$ contains only principles which are as uncontroversial as logical principles ever can be, and Enthymeme is an easily-grasped and natural extension. See Sec. 2.10.7 for further details.

The major difference between (positive) intuitionistic logic and relevant logic (at least at the level of $R_+$) is expressed algebraically by the forms of Residuation
they support. Intuitionistic residuation corresponds to:

- \( A \& B \rightarrow C \vdash B \rightarrow A \rightarrow C \)

See [Cur77, ??] for an algebraic treatment of intuitionism. With this principle \( B \rightarrow A \rightarrow B \) follows immediately from basic principles for \&. Readers will be able to relate the alternative forms of Residuation back to the different versions of the Ramsey Test in Chapter 1. The fusion form of Residuation given above is one of the main reasons why fusion itself is interesting to relevant logicians, providing as it does a way of expressing the what is taken to be the failing in the intuitionistic form (i.e., that \( A \) and \( B \) should both be relevant somehow to \( C \) for the conclusion to follow). Similarly, the Enthymeme principle provides the major interest for fusion in our systems; if we were to take instead the rejected form

\[
A \& (A \& B \rightarrow C) \vdash B \rightarrow C
\]

then our construction would deliver intuitionistic rather than relevant logic.

### 2.5 Permission

Here we introduce another extension to the language with the connective \( \sim \), 'permission'. \( A \sim B \) is to be read as: 'The truth of \( A \) permits the truth of \( B \)'. Permission is intended to provide a dual to implication in the same way that conjunction and disjunction are dual to each other. Although it has some independent interest, we will mainly be interested in permission as an aid to the formation of negation principles for implication, and it will be factored out of the fragment we implement in the logic programming system described in Chapter 4.

Our reading of \( A \rightarrow B \) has a quality of necessity: 'If \( A \) is true then \( B \) must therefore be true'. Dually, permission has a flavour of possibility: 'If \( A \) is true, then \( B \) might be true (even so)' is a reading which we want to fall within the range covered by our interpretation of \( A \sim B \). There is no suggestion that \( A \) has any part in making \( B \) true - permission has more the character of a statement of independence; 'The
truth of $A$ does not rule out the possibility of the truth of $B'$. These kind of motivations will lead naturally to the equivalence of 'not $A \rightarrow B$' and 'A~ not B' in our negation systems. To this extent, permission is similar to the consistency connective of [LL32, VI.4].

In the logics generated by our stronger systems with negation, fusion and permission will be interdefinable (Sec 3.8). See [AB75, pp345-6] for a discussion of the combined consistency/conjunctive character of fusion in standard relevant formulations. There is at least some motivational benefits in separating out these characteristics, which are related but clearly different. (Our view of the nature of the relationship is embodied in the $F$ principle later in this section.)

In the chemical model, implication corresponded to substances which serve to establish a reaction which uses the antecedent to produce the consequent. Permission, on the other hand, is something which serves to block any process involving the antecedent which leads to the destruction of the consequent.

We will start with with these Monotonicity principles, which seem straightforward:

Monotonicity

\[ M\sim_1 \quad B \Rightarrow C \vdash A\sim B \Rightarrow A\sim C \]
\[ M\sim_2 \quad A\Rightarrow B \vdash A\sim C \Rightarrow B\sim C \]

If $A$ permits $B$ then $A$ permits anything which follows from $B$; if $A$ permits $C$ then anything which follows from $A$ will permit $C$ also (for if such a $B$ were to rule out $C$, then presumably $A$ would also). Note the asymmetry compared to $M\rightarrow_1$ and $M\rightarrow_2$: $M\sim_1$ follows the ordering of $M\rightarrow_1$, but $M\sim_2$ reverses the ordering of $M\rightarrow_2$.

To motivate the conditions relating permission and $\&$ and $\lor$, consider the following arguments:

- If $A$ does not permit $B$, and $A$ does not permit $C$, then $A$ will not permit 'B or C'. So if $A$ permits 'B or C' then $A$ permits $B$, or $A$ permits $C$. 
• If $A$ does not permit $C$, and $B$ also does not permit $C$, then ‘$A$ or $B$’ will not permit $C$. So if ‘$A$ or $B$’ permits $C$, $A$ permits $C$ or $B$ permits $C$.

Translated, these arguments give us:

**Permission Strengthening**

$P_{\lor 1} \; A \rightarrow B \lor C \Rightarrow (A \rightarrow B) \lor (A \rightarrow C)$

$P_{\lor 2} \; A \lor B \rightarrow C \Rightarrow (A \rightarrow C) \lor (B \rightarrow C)$

In virtue of $M \rightarrow 1$ and $M \rightarrow 2$ the converse of each of these principles also holds. We can also derive the following:

• $A \& C \rightarrow B \lor D \Rightarrow (A \rightarrow B) \lor (C \rightarrow D)$

• $A \lor C \rightarrow B \& D \Rightarrow (A \rightarrow B) \lor (C \rightarrow D)$

• $A \& C \rightarrow B \& D \Rightarrow (A \rightarrow B) \& (C \rightarrow D)$

By analogy with $P_{\lor 1}$ and $P_{\lor 2}$ we can take the following principles for fission and permission:

$P_{+1} \; A \rightarrow B \lor C \Rightarrow (A \rightarrow B) \lor (A \rightarrow C)$

$P_{+2} \; A \lor B \rightarrow C \Rightarrow (A \rightarrow C) \lor (B \rightarrow C)$

As was the case for implication strengthening with respect to intensional conjunction and disjunction, we have some plausible alternatives:

• $A \circ C \rightarrow B \lor D \Rightarrow (A \rightarrow B) \lor (C \rightarrow D)$

• $A \lor C \rightarrow B \circ D \Rightarrow (A \rightarrow B) \lor (C \rightarrow D)$

• $A \circ C \rightarrow B \circ D \Rightarrow (A \rightarrow B) \circ (C \rightarrow D)$

$P_{+1}$ and $P_{+2}$ are derivable from these. As before, we choose the simpler principles in the absence of any stronger intuitions which might discriminate between them.
We should point out that we definitely do not want any of the following principles for permission:

- \((A \sim C) \& (B \sim C) \Rightarrow A \& B \sim C\)
- \((A \sim B) \& (A \sim C) \Rightarrow A \sim B \& C\)

The reasons for rejecting these are straightforward: the conjunction of \(A\) and \(B\) might not permit something permitted by both \(A\) and \(B\) separately, and \(A\) might permit both \(B\) and \(C\) without permitting their conjunction. In each case, the argument relies on the intuition that when we mix things together, we generally get more than we started with.

A more subtle point is raised by the consideration of whether we would want

\[ C \quad A \rightarrow B \Rightarrow A \sim B \]

that is, whether we should take permission as weaker than implication. Although the principle is not totally implausible, we think that it should not be taken as an entailment. It seems consistent for \(A\) at the same time to be sufficient for \(B\) and to rule it out (standardly, we would conclude that \(A\) is 'impossible' in these circumstances), so the meaning of the permission would not seem to be included in the implication. Were we to include this principle, \(\neg(A \rightarrow B) \iff A \sim \neg B\) would be derivable in our negation systems, counter to this intuition. Our 'permission' is very similar to the 'compatibility' of Connexivist logics for which \(C\) and this derived principle both hold; see McCall in [AB75, §29.8] for a good presentation of one species of Connexivism and [SMPB82, pp82-95] for a critique.

However we will consider systems which add the following principle:

**Permission Ordering**

\[ F \& A \& B \Rightarrow A \sim B \]

or in fusion form:

\[ F \circ A \circ B \Rightarrow A \sim B \]
From $A$ and $B$ we can infer that the truth of $A$ permits the truth of $B$. On the chemical analogy, the tightly bonded molecules formed from the contents of $A$ and $B$ constitute something which blocks any reaction involving $A$ which acts to destroy $B$. The effect of this will be to give $A\land \neg B \Rightarrow \neg (A \rightarrow B)$ (and $A\lor \neg B \Rightarrow \neg (A \rightarrow B)$) in the systems with negation of the next chapter, principles which we take to be highly plausible.

2.6 Positive Systems with Quantifiers

We define free and bound occurrences of variables in a wff as usual, and we let $A\{s/x\}$ denote the wff obtained from $A$ by consistently replacing every free occurrence of $x$ in $A$ by the term $s$. Where the variables occurring free in $A$ are $x, y, z \ldots$ then $\forall A$ denotes $(\forall x)(\forall y)(\forall z)\ldots A$, the universal closure of $A$; $\exists A$ denotes $(\exists x)(\exists y)(\exists z)\ldots A$, the existential closure.

In our systems, universally and existentially quantified formulas are best regarded as short-hand for infinite conjunctions and disjunctions of quantifier-free formulas: that is, $(\forall x)A$ is taken as equivalent to $A\{s_1/x\} \& A\{s_2/x\} \& \ldots$, and $(\exists x)A$ as equivalent to $A\{s_1/x\} \lor A\{s_2/x\} \ldots$, where $s_1, s_2, \ldots$ are all the terms in $L$. So our principles for quantifiers will be simple generalizations of propositional principles.

- Axioms.

**Weakening**

$W\forall (\forall x)A \Rightarrow A\{s/x\}$ (for all $s$).

$W\exists A\{s/x\} \Rightarrow (\exists x)(A)$ (for all $s$).

For the next group of axioms, we assume in each case that $x$ does not occur free in $A$.

**Idempotence**

$I\forall A \Rightarrow (\forall x)A$.

$I\exists (\exists x)A \Rightarrow A$. 
CHAPTER 2. A FORMULATION OF RELEVANCE

Associativity

\[ \forall x (A \lor B) \rightarrow A \lor (\forall x)B. \]
\[ \forall x (A + B) \rightarrow A + (\forall x)B. \]

Distribution

\[ D \& \exists A & (\exists x)B \rightarrow (\exists x)(A & B). \]
\[ D \circ \exists A \circ (\exists x)B \rightarrow (\exists x)(A \circ B). \]

Strengthening

\[ S \forall (\forall x)(A \rightarrow B) \rightarrow A \rightarrow (\forall x)B. \]
\[ S \exists (\forall x)(B \rightarrow A) \rightarrow (\exists x)B \rightarrow A. \]

Permission Strengthening

\[ P \exists_1 A \sim (\exists x)B \rightarrow (\exists x)(A \sim B). \]
\[ P \exists_2 (\exists x)B \sim A \rightarrow (\exists x)(B \sim A). \]

- Rules.

Rules

\[ M \forall A \rightarrow B \vdash (\forall x)A \rightarrow (\forall x)B. \]
\[ M \exists A \rightarrow B \vdash (\exists x)A \rightarrow (\exists x)B. \]

2.7 Summary of Positive Principles

For ease of reference, we give here a complete list of the principles discussed so far.

To make things more compact, and to emphasize relationships between the properties of the adjunctive connectives, we will adopt the following shorthand:

- ‘⊕’ may stand for either of the disjunctive connectives, + and ∨.
- ‘⊗’ may stand for either of the conjunctions, ° and &.
- ‘•’ may stand for any of °, &, + and ∨.
In the quantifier principles marked with an "*", \( x \) may not occur free in \( A \).

- Axioms.

  **Idempotence**
  
  \[ I \otimes A \Rightarrow A \otimes A. \]
  
  \[ I \oplus A \oplus A \Rightarrow A. \]
  
  \[ I \forall A \Rightarrow (\forall x)A (*) . \]
  
  \[ I \exists (\exists x)A \Rightarrow A (*) . \]

  **Commutativity**
  
  \[ C \otimes A \circ B \Rightarrow B \circ A \]

  **Associativity**
  
  \[ A \circ A \circ B \circ C \Rightarrow A \circ (B \circ C) \]
  
  \[ A + \& (A+B) & (A+C) \Rightarrow A+B & C \]
  
  \[ A \oplus \forall (\forall x)(A \oplus B) \Rightarrow A \oplus (\forall x)B (*) \]

  **Distribution**
  
  \[ D \otimes \forall A \otimes (B \lor C) \Rightarrow (A \otimes B) \lor (A \otimes C) \]
  
  \[ D \circ \exists A \circ (\exists x)B \Rightarrow (\exists x)(A \otimes B) (*) \]

  **Weakening**
  
  \[ W \& A \& B \Rightarrow A \]
  
  \[ W \lor A \Rightarrow A \lor B \]
  
  \[ W \forall (\forall x)A \Rightarrow A\{s/x\} \text{ (all } s\). \]
  
  \[ W \exists A\{s/x\} \Rightarrow (\exists x)A \text{ (all } s\). \]

  **Strengthening**
  
  \[ S \otimes (A \rightarrow B) \otimes (A \rightarrow C) \Rightarrow A \rightarrow B \otimes C \]
  
  \[ S \oplus (A \rightarrow C) \otimes (B \rightarrow C) \Rightarrow A \oplus B \rightarrow C \text{ (where } \oplus \text{ is the 'dual' of } \otimes \text{).} \]
  
  \[ S \lor (\forall x)(A \rightarrow B) \Rightarrow A \rightarrow (\forall x)B (*) \]
CHAPTER 2. A FORMULATION OF RELEVANCE

\[ \exists (\forall x)(B \rightarrow A) \Rightarrow (\exists x)B \rightarrow A \quad (*) \]

\[ P \oplus_1 A \sim B \oplus C \Rightarrow (A \sim B) \oplus (A \sim C) \]

\[ P \oplus_2 A \oplus B \sim C \Rightarrow (A \sim C) \oplus (B \sim C) \]

\[ P \exists_1 A \sim (\exists x)B \Rightarrow (\exists x)A \sim B \quad (*) \]

\[ P \exists_2 (\exists x)B \sim A \Rightarrow (\exists x)(B \sim A) \quad (*) \]

Transitivity

\[ T \otimes (A \rightarrow B) \otimes (B \rightarrow C) \Rightarrow A \rightarrow C \]

Detachment

\[ D \otimes A \otimes (A \rightarrow B) \Rightarrow B \]

Enthymeme

\[ E \ A \circ (A \circ B \rightarrow C) \Rightarrow B \rightarrow C \]

Permission Ordering

\[ F \otimes A \otimes B \Rightarrow A \sim B \]

• Rules.

Transitivity

\[ T \Rightarrow A \Rightarrow B, B \Rightarrow C \vdash A \Rightarrow C \]

Monotonicity

\[ M \bullet A \Rightarrow B \vdash A \bullet C \Rightarrow B \bullet C \]

\[ M \forall A \Rightarrow B \vdash (\forall x)A \Rightarrow (\forall x)B \]

\[ M \exists A \Rightarrow B \vdash (\exists x)A \Rightarrow (\exists x)B \]

\[ M \rightarrow_1 A \Rightarrow B \vdash B \rightarrow C \Rightarrow A \rightarrow C \]

\[ M \rightarrow_2 B \Rightarrow C \vdash A \rightarrow B \Rightarrow A \rightarrow C \]

\[ M \sim_1 A \Rightarrow B \vdash A \sim C \Rightarrow B \sim C \]

\[ M \sim_2 B \Rightarrow C \vdash A \sim B \Rightarrow A \sim C \]
CHAPTER 2. A FORMULATION OF RELEVANCE

2.8 Hierarchy of Positive Systems

Our basic propositional system, which we will call \( \mathcal{E}_+ \), is restricted to the part of \( L \) which has just the connectives \& and \( \lor \). The principles we take for \( \mathcal{E}_+ \) are as follows:

- Axioms: \( W\& \), \( I\& \), \( C\& \), \( A\& \), \( W\lor \), \( IV \), \( CV \), \( AV \), \( D\&\lor \), \( S\& \), \( SV \).
- Rules: \( T\Rightarrow \), \( M\& \), \( M\lor \), \( M\rightarrow_1 \), \( M\rightarrow_2 \).

Our basic quantified system, called \( \mathcal{QE}_+ \), adds the universal and existential quantifiers with the following principles:

- Axioms: \( W\forall \), \( IV \), \( A\forall \), \( S\forall \), \( W\exists \), \( I\exists \), \( S\exists \), \( D\&\exists \).
- Rules: \( M\forall \), \( M\exists \).

Results and definitions given for propositional systems are to be understood as applying equally to their quantified counterparts, except where otherwise noted.

Extensions to the basic systems fall into two classes: conservative linguistic extensions, which add \( \circ \), \(+\) or \( \sim \) (or some combination) to the basic system, and non-conservative extensions which add principles giving new properties for existing linguistic elements. These extensions are given as follows:

- Linguistic extensions. These are restricted by the requirement that whenever we have \(+\) we also have \( \circ \) (although \( \circ \) without \(+\) is allowed). Thus we may have the following combinations: \( \circ \) by itself; \( \sim \) by itself; \( \circ \) and \(+\); \( \circ \) and \( \sim \); \( \circ \) and \(+\) and \( \sim \). These will be indicated by superscripts: respectively, \( \circ \), \( \sim \), \(+\), \( \circ \sim \), \(+\sim \). (Since we can never have \(+\) without \( \circ \), it is enough to indicate the presence of the former). So \( \mathcal{E}_+^{\circ \sim} \) adds each of \( \circ \), \(+\) and \( \sim \) to the language of the basic propositional system, and \( \mathcal{QE}_+^{\circ \sim} \) is the corresponding extension to the basic quantified system. The additional principles for each extension are as follows:
  - For \( \circ \) by itself: \( I\circ \), \( C\circ \), \( A\circ \), \( D\circ\lor \), \( S\circ \), \( M\circ \). Quantified systems also add \( D\circ\exists \).
- For $\vdash$ by itself: $P\lor_1, P\lor_2, M\vdash_1, M\vdash_2$. Quantified systems also add $P\exists_1$ and $P\exists_2$.
- For $+:$ as for $\circ$, plus $I+$, $C+$, $A+$, $D+$, $A+\&$, $S+$, $M+$.
  Quantified systems also add $A+\forall$.
- For $\circ\vdash$: combination of the principles for $\circ$ and $\vdash$.
- For $\vdash\vdash$: as for $\circ\vdash$, plus the principles for $+$, plus $P+1$ and $P+2$.

**Non-conservative extensions.** These add Transitivity, Detachment, Enthymeme or Permission Ordering principles, or some combination, to the underlying basic system or a linguistic extension to it. They are indicated by a superscript: $T$ for Transitivity, $D$ for Detachment, $E$ for Enthymeme, and $F$ for permission ordering. Thus $E_{\circ\vdash}^T E$ is the system which adds Transitivity and Enthymeme to $E_{\circ\vdash}$. The reason for not including these in the basic systems fall into two classes: either there are instructive formal distinctions between systems with the principles and those without ($T$ and $D$), or the principle serves to give a extended sense to implication or permission ($E$ and $F$) which we take to be natural without superseeding the basic sense. The principles appropriate to the extensions are as follows:

- **T** Where $\circ$ occurs in the superscript, add $T\circ$, otherwise add $T\&$ (note that $T\&$ follows from $T\circ$ in any system with fusion).
- **D** Where $\circ$ occurs in the superscript, add $D\circ$, otherwise $D\&$ (again, $D\&$ follows from $D\circ$ in any fusion system).
- **E** There is no form of Enthymeme without $\circ$, so this can only occur together with $\circ$ or $+$ in a superscript, in which case add the $E$ principle.
- **F** This only makes sense if $\vdash$ is in the superscript. Where $\circ$ or $+$ occurs in the superscript, add $F\circ$, otherwise $F\&$ (which follows from $F\circ$ in any system with fusion).

We will often refer to the ‘language of $E_{\circ}$’: by this we mean the fragment of $L$ with the basic connectives $\&, \lor$ and $\rightarrow$, plus whatever additional connectives are indicated by the superscript $\alpha$; similarly for quantified systems (which take $\forall$ and
3 as basic also). So for example the language of $\mathcal{E}_{+}^{TDE}$ is the fragment of $L$ with the basic connectives plus $+$ and $\circ$.

We also have restricted systems $\mathcal{E}_{+z}^{\alpha}$. In each case this is the system generated by just principles of $\mathcal{E}_{+}^{\alpha}$ which do not contain any occurrences of $\rightarrow$, with the understanding that the vocabulary of $\mathcal{E}_{+z}^{\alpha}$ does not contain $\rightarrow$.

We will be quite loose in our treatment of the superscript notation, using $\mathcal{E}_{+}^{\alpha}$ for example to denote the $'+$ extension to a system $\mathcal{E}_{+}^{\alpha}$ for which it is understood that $+$ does not occur in $\alpha$. We will treat the superscript sometimes as a set, and sometimes as a list, whichever is most convenient for the current purpose.

2.9 Some Properties of Positive Entailment Systems

2.9.1 Weak Relevance

All of our propositional systems have the weak relevance or variable sharing property:

Theorem 1 If $\vdash_{\mathcal{E}_{+}^{\alpha}} A \rightarrow B$ then there is some atom $p$ which occurs in both $A$ and $B$.

The proof of this uses a simple two-valued valuation on formulas:

- $v(p) \in \{0, 1\}$ for all atoms $p$.
- $v(A \& B) = v(A \circ B) = v(A \rightarrow B) = \min(v(A), v(B))$.
- $v(A \lor B) = v(A + B) = \max(v(A), v(B))$.
- $v(A \rightarrow B) = v(B)$.

It is easy if rather tedious to show that whenever $A \rightarrow B$ is provable in $\mathcal{E}_{+}^{\alpha}$ (for any $\alpha$), then $v(B) = 1$ whenever $v(A) = 1$. Clearly if $v(p) = 0$ for every atom
occurring in $A$, $v(A) = 0$; conversely if $v(p) = 1$ for all these atoms then $v(A) = 1$. So if $A$ and $B$ share no atoms, we can set $v(B) = 0$ by setting all of it atoms to 0, and similarly we can make $v(A) = 1$, and hence $A \Rightarrow B$ will not be provable in $\mathcal{E}_+^*$.

2.9.2 Tautological Entailments

Theorem 2 $A \Rightarrow B$ is provable in $\mathcal{E}_+^2$ iff $A \rightarrow B$ is a theorem of the logic $R_{+fde}$

$R_{+fde}$ is the positive fragment of the 'tautological entailments' logic called $E_{fde}$ in [AB75, §15.2]), and has the following axiomatization:

Axioms

A1 $A \rightarrow A$

A2 $A \& B \rightarrow A$, $A \& B \rightarrow B$

A3 $A \rightarrow A \lor B$, $B \rightarrow A \lor B$

A4 $A \&(B \lor C) \rightarrow (A \& B) \lor C$

Rules

R1 $A \rightarrow B$, $B \rightarrow C \vdash A \rightarrow C$

R2 $A \rightarrow B$, $A \rightarrow C \vdash A \rightarrow B \& C$

R3 $A \rightarrow C$, $B \rightarrow C \vdash A \lor B \rightarrow C$

All formulas in this system have the form $A \rightarrow B$, where it is understood that $A$ and $B$ are built up from atomic constituents using just $\&$ and $\lor$.

2.9.3 Replacements

Theorem 3 (Replacement of Equivalents) For any positive entailment system, if $A \leftrightarrow B$ is provable in the system, and $D$ is like $C$ except for substituting an occurrence of $B$ for some occurrence of $A$, then $C \leftrightarrow D$ is provable also.
The proof of this is the usual induction on the complexity of formulas, making use of the Monotonicity principles.

**Theorem 4 (Atom Replacement)** If $A \Rightarrow B$ is provable in $\mathcal{E}_+^\alpha$, then so is $A' \Rightarrow B'$ where $A'$ and $B'$ result from $A$ and $B$ by replacing every occurrence of some atom $p$ by another atom $q$.

The proof is by induction on the construction of $\mathcal{E}_+^\alpha$. As a corollary, whenever $A \Rightarrow B$ is provable in $\mathcal{E}_+^\alpha$ then so is $A\{s/x\} \Rightarrow B\{s/x\}$ for any term $s$, since $A\{s/x\}$ and $B\{s/x\}$ differ from $A$ and $B$ only in substituting atoms $p\{s/x\}$ for atoms $p$.

**Definition 2 (Positive and Negative Parts)** Occurrences of formulas within other formulas are divided into positive and negative occurrences as follows:

- A formula $A$ is a positive occurrence in itself.

- A positive (negative) occurrence of $B$ in $A$ is a positive (negative) occurrence in $A \oplus C$, $C \oplus A$ or $QA$, where $\oplus$ is any of $\&$, $\lor$, $+$, $\circ$ or $\neg$, and $Q$ is any quantifier.

- A positive (negative) occurrence of $B$ in $A$ is a negative (positive) occurrence in $A \rightarrow C$, and a positive (negative) occurrence in $C \rightarrow A$.

- Additionally, where the major connective of a formula occurring positively (negatively) is $\oplus$, we will say that the occurrence of $\oplus$ is positive (negative).

**Theorem 5 (Directed Replacement)** Let $D$ result from $C$ by substituting $B$ for a positive (negative) occurrence of $A$. Then if $A \Rightarrow B$ ($B \Rightarrow A$) is provable in $\mathcal{E}_+^\alpha$, so is $C \Rightarrow D$.

**Proof** By induction on the complexity of the occurrence of $A$ in $D$, defined as follows:

- The occurrence of $A$ in itself has complexity 0.
• If the complexity of an occurrence of \( A \) in \( B \) is \( n \), then the complexity of the corresponding occurrence of \( A \) in \( B \cdot C, C \cdot B \) and \( QB \) is \( n + 1 \), where * is any binary connective and \( Q \) is any quantifier.

The details of the proof are straightforward, relying on the Monotonicity principles for each connective.

### 2.9.4 Reductions

**Theorem 6** If \( A \Rightarrow B \) is provable in \( \mathcal{E}_+^\beta \), and \( E \not\in \beta \) and \( \beta \) is either \( \alpha \Rightarrow \) or \( \alpha \Leftrightarrow \), then \( A' \Rightarrow B' \) is provable in \( \mathcal{E}_+^\alpha \), where \( A' \) and \( B' \) result from \( A \) and \( B \) respectively by replacing every occurrence of \( \alpha \) by \( \& \) and every occurrence of \( + \) by \( \lor \).

**Proof** A straightforward induction on the construction of \( \mathcal{E}_+^\alpha \):

- **Basis.** Inspection will confirm that for every axiom of \( \mathcal{E}_+^\beta \), \( A' \Rightarrow B' \) is an axiom of \( \mathcal{E}_+^\alpha \).

- **Inductive step.** This is straightforward, given that the rules of \( \mathcal{E}_+^\beta \) are just \( \Rightarrow \text{-}\Rightarrow \) and Monotonicity principles, and clearly each has its appropriate counterpart in \( \mathcal{E}_+^\alpha \).

Note that Enthymeme breaks this correspondence; \( A \& (A \& B \Rightarrow C) \Rightarrow B \Rightarrow C \) will not be demonstrable in \( \mathcal{E}_+^\alpha \).

**Theorem 7** If \( A \Rightarrow B \) is provable in \( \mathcal{E}_+^\beta \), where \( \beta \) is either \( \alpha \Leftrightarrow \) or \( \alpha \Rightarrow \neg \), then \( A' \Rightarrow B' \) is provable in \( \mathcal{E}_+^\alpha \), where \( A' \) and \( B' \) result from \( A \) and \( B \) by replacing every occurrence of \( C \Leftrightarrow D \) by \( C \Rightarrow D \) where \( \Rightarrow \) is \( \alpha \) if \( + \) or \( \circ \) occur in \( \alpha \); otherwise it is \( \& \).

The proof of this theorem uses the normal induction on the construction of \( \mathcal{E}_+^\beta \):

- **Basis.** If \( A \Rightarrow B \) is an instance of an axiom-form common to \( \mathcal{E}_+^\beta \) and \( \mathcal{E}_+^\alpha \) then clearly \( A' \Rightarrow B' \) is an axiom of \( \mathcal{E}_+^\alpha \). Otherwise it is an instance of one of
the axioms-forms in which permission is explicit. The translation of these instances will in each case give a result which is either an axiom of $E^\alpha_+$ or derivable using simple moves. The $F\otimes$ principle, for example, becomes $A \otimes B \Rightarrow A \otimes B$ which is of course derivable.

- As far as the $E^\beta_+$ derivation rules go, we need only consider the Monotonicity principles for permission, which reduce under translation either to applications of $M\otimes$ or combinations of $C\otimes$ and $M\otimes$.

**Theorem 8** If $A\Rightarrow B$ is provable in $QE^\alpha_+$ then $A'\Rightarrow B'$ is provable in $E^\alpha_+$, where $A'$ and $B'$ result from $A$ and $B$ by replacing every occurrence of $(\forall x)C$ by the conjunction of all instances of $C\{s/x\}$, for every term $s$ in the language, and every occurrence of $(\exists x)C$ by the disjunction of all these instances.

**Proof** Inspection will confirm that the statement is clearly true for all of the quantifier axioms (noting in $\forall \alpha$ and $\exists \alpha$ that $A\{s/x\}$ is just $A$). That $M\forall$ and $M\exists$ preserve the property follows from Th. 4; if $A\Rightarrow B$ is provable in $E^\alpha_+$ then so is $A\{s/x\} \Rightarrow B\{s/x\}$ for any $s$, and the result follows by applications of Weakening and Monotonicity.

We give the following result here for future reference; it will be important in determining the fragment computed by our logic programming system; see Sec. 4.3 for further details.

**Theorem 9** If $A$ contains no positive occurrences of wffs of the form $C\rightarrow D$, and $B$ no negative occurrences, and $P, F \notin \alpha$, then $A\Rightarrow B$ is provable in $E^{\alpha P F}_+$ iff $A'\Rightarrow B'$ is provable in $E^\alpha_+$, where $A'$ and $B'$ result from $A$ and $B$ by replacing all occurrences of $\rightarrow$ by $\circ$, if $+$ or $\circ$ is in $\alpha$, or otherwise by $\&$.

Left to right follows by Th. 7. For the other direction, assume that $A$ and $B$ are as in the statement of the theorem, and that $A'\Rightarrow B'$ is provable in $E^\alpha_+$; then it will be provable in $E^{\alpha P F}_+$ also. The result will follow by Th. 5 if we can show that $A\Rightarrow A'$ and $B'\Rightarrow B$ are both provable in $E^{\alpha P F}_+$. But this is indeed the case: $A'$ results from $A$ by replacing negative occurrences of $C\rightarrow D$ by $C \circ D$ (or $C \& D$), and $A\Rightarrow A'$ follows from Th. 5 via the $F$ principle; dually for $B'\Rightarrow B$. 

2.9.5 Conservative Extensions

An entailment system $\mathcal{E}^\alpha$ is a conservative extension of another system $\mathcal{E}^\beta$ just in case whatever is provable in $\mathcal{E}^\beta$ is also provable in $\mathcal{E}^\alpha$, and whenever $A \Rightarrow B$ is provable in $\mathcal{E}^\alpha$ and $A$ and $B$ are in the vocabulary of the smaller system $\mathcal{E}^\beta$, then $A \Rightarrow B$ is provable in $\mathcal{E}^\beta$ also. So $\mathcal{E}^\alpha$ is a conservative extension of $\mathcal{E}^\beta$ if the only new things it adds are entailments which are not expressible in the vocabulary of $\mathcal{E}^\beta$.

For some of the results here we will use valuations based on the on the algebra of formulas induced by our entailment systems (the method is based on the one Meyer uses in [AB75, §24.4.2]). For every entailment system $\mathcal{E}^\alpha$ the corresponding formula algebra is $A^\alpha = \langle M, \preceq, O \rangle$. $M$ is the set of equivalence classes induced by the $\mathcal{E}^\alpha \Rightarrow$ relation: that is the set of elements $\epsilon(A) = \{ B : \mathcal{E}^\alpha A \leftrightarrow B \}$, for every $A$ in the language of $\mathcal{E}^\alpha$. The ordering $\preceq$ is a binary relation on elements of $M$ defined as follows: $\epsilon(A) \preceq \epsilon(B)$ iff $A \Rightarrow B$ is provable in $\mathcal{E}^\alpha_+$. Finally, $O$ is a set of operations on elements of $M$, corresponding to the connectives in the vocabulary of $\mathcal{E}^\alpha$. We use the same symbols for the algebraic operations as the connectives, and for each such connective $\oplus$ the corresponding operation is defined as: $\epsilon(A) \oplus \epsilon(B) = \epsilon(A \oplus B)$; or for the quantifiers: $(\forall x)\epsilon(A) = \epsilon((\forall x)A)$ and $(\exists x)\epsilon(A) = \epsilon((\exists x)A)$.

A valuation for $\mathcal{E}^\alpha$ in an algebra $\langle M, \preceq, O \rangle$ is just a mapping $v(A)$ from formulas in the vocabulary of $\mathcal{E}^\alpha$ into elements of $M$. We say that the valuation is a model for $\mathcal{E}^\alpha$ iff $v(A) \preceq v(B)$ whenever $A \Rightarrow B$ is provable in $\mathcal{E}^\alpha$, and that it is characteristic for $\mathcal{E}^\alpha$ iff the converse of this also holds. For each rule system $\mathcal{E}^\alpha$, we use $A^\alpha$ to define the canonical valuation as follows: $v^\alpha(A) = \epsilon(A)$. Clearly, the canonical valuation $v^\alpha$ is characteristic for $\mathcal{E}^\alpha$: $v^\alpha(A) \preceq v^\alpha(B)$ iff $\epsilon(A) \preceq \epsilon(B)$ iff $\vdash \mathcal{E}^\alpha A \Rightarrow B$.

To demonstrate that $\mathcal{E}^\alpha$ is a conservative extension of $\mathcal{E}^\beta$ we expand $A^\beta = \langle M, \preceq, O \rangle$ to $\langle M^*, \preceq^*, O \rangle$ in such a way that $M \subseteq M^*$ and $a \preceq b$ iff $a \preceq^* b$ for all $a, b \in M$. We define a mapping $v^*$ from sentences in the vocabulary of the larger system $\mathcal{E}^\alpha_+$ to elements of $M^*$ which extends the canonical valuation for
\( E_+^\beta \) in the sense that \( v^\ast(A) = v^\beta(A) \) for all sentences \( A \) in the language of the smaller system \( E_+^\beta \). Thus we will have the following, for all \( A \) and \( B \) in the vocabulary of \( E_+^\beta \): \( \vdash E_+^\beta A \Rightarrow B \) iff \( v^\beta(A) \leq v^\beta(B) \) iff \( v^\ast(A) \leq^\ast v^\ast(B) \). Thus the valuation is characteristic for \( E_+^\beta \) in its restriction to the \( E_+^\beta \) vocabulary. Additionally, the expanded algebra and valuation are constructed in such a way that \( v^\ast(A) \leq^\ast v^\ast(B) \) whenever \( A \Rightarrow B \) is provable in \( E^\alpha \), for \( A \) and \( B \) in the larger systems vocabulary; thus the extended valuation is a model for \( E_+^\alpha \). If we succeed in constructing extended algebras and valuations with these properties, then we have indeed demonstrated that \( E_+^\alpha \) is a conservative extension of \( E_+^\beta \). For let \( A \) and \( B \) be sentences in the vocabulary of the smaller system, and assume that \( A \Rightarrow B \) is not provable in it. Then \( v^\beta(A) \not\leq v^\beta(B) \) (since \( v^\beta \) is characteristic for \( E_+^\beta \)) and so \( v^\ast(A) \not\leq^\ast v^\ast(B) \) (by construction). Since we have constructed the extensions in such a way that \( v^\ast \) is a model for \( E_+^\alpha \) we can conclude that \( A \Rightarrow B \) is not provable in the larger system either.

The details of the constructions are as follows. Let \( M^* = M \cup \{ \top, \bot \} \), and make the following additions to \( \leq \) to give \( \leq^\ast \) (for all \( a \in M^* \)):

- \( \bot \leq^\ast a \).
- \( a \leq^\ast \top \).

So \( \leq^\ast \) extends \( \leq \) just by adding a top element and a bottom element, and clearly we have met our first requirement, that \( a \leq^\ast b \) iff \( a \leq b \) for all \( a, b \in M \). Note that \( \leq \) will be transitive for the formula algebras corresponding to each of our systems, and \( \leq^\ast \) retains this characteristic. The operations of \( O \) are extended to the new elements of \( M^* \). For all \( a \in M^* \) (whenever the indicated operation is in \( O \)):

- \( a + T = T + a = a \lor T = T \lor a = a \rightarrow T = \bot \rightarrow a = (\forall x)T = (\exists x)T = T \).
- \( a \& \bot = \bot \& a = a \& \bot = \bot \& a = a \rightarrow \bot = \bot \rightarrow a = (\forall x)\bot = (\exists x)\bot = \bot \).
- \( a \& T = T \& a = a \lor \bot = \bot \lor a = a \).
- If \( a \neq \bot \), \( a \rightarrow \bot = \bot \) and \( a \circ T = T \circ a = a \rightarrow T = T \rightarrow a = T \).
CHAPTER 2. A FORMULATION OF RELEVANCE

- If \( a \neq T, T \rightarrow a = \perp \) and \( a + \perp = \perp + a = \perp \).

For the extended valuation \( v^* \) we let \( v^*(A) = v^\beta(A) \) for all \( A \) in the vocabulary of the smaller system \( \mathcal{E}^\beta_+ \), meeting our first requirement on the extended valuation.

For the connectives \( \otimes \) introduced by the larger system \( \mathcal{E}^\alpha_+ \), we extend \( v^\beta \) in one of two ways:

**Top Extension** \( v^*(A \otimes B) = T \), for all \( A,B \).

**Bottom Extension** \( v^*(A \otimes B) = \perp \), for all \( A,B \).

Clearly the valuations constructed in this way meet our requirement that they be characteristic for the smaller system, in their restriction to the smaller system’s vocabulary. What remains to be shown in each case is that the extensions are models for the larger system. We can split this task into two parts; first of all showing that \( v^*(A) \preceq^* v^*(B) \) whenever \( A \Rightarrow B \) is provable using just the principles of the smaller system \( \mathcal{E}^\beta_+ \), although \( A \) and \( B \) may themselves contain connectives introduced in the vocabulary in the larger system, and secondly showing that the new axioms and rules introduced in the larger system preserve this property. The following lemmas will establish the first half of this:

**Lemma 1** For every instance \( A \Rightarrow B \) in the language of \( \mathcal{E}^\alpha \) of every axiom form of \( \mathcal{E}^\beta_+ \), \( v^*(A) \preceq^* v^*(B) \).

**Proof** To illustrate, take \( W \&: \) we want to show that \( a \& b \preceq^* a \) for all \( a, b \in \mathcal{M}^* \), whenever \( a \& b \preceq a \) for all \( a, b \in \mathcal{M} \). Since the restriction of \( \preceq^* \) to \( \mathcal{M} \) is just \( \preceq \), we need only check for the case where at least one of \( a \) and \( b \) is either \( T \) or \( \perp \). If \( a = T \) then we are done (since \( c \preceq^* T \) for all \( c \)). If \( a = \perp \) then \( a \& b = \perp \) also, and of course \( \perp \preceq^* \perp \). If \( b = T \) then \( a \& b = a \), and if \( b = \perp \) then \( a \& b = \perp \), so the relation holds for these assignments also. The proof for each axiom proceeds by same kind of truth-tables method, and was actually carried out by a small program written for the purpose. Given an input \( f(a_1 \ldots a_m) \preceq^* g(b_1 \ldots b_n) \) for some composition of operators \( f \) and \( g \) corresponding to one of the axiom forms of the smaller system, the program works through the possibilities for each
CHAPTER 2. A FORMULATION OF RELEVANCE

\[ a_1 \ldots a_m, b_1 \ldots b_n \] (each must be either an element of \( M \), or \( T \), or \( \bot \)) and reports the cases for which \( f(a_1 \ldots a_m) \neq \bot \) and \( g(b_1 \ldots b_n) \neq T \). In each such case, as it turns out, we have \( f(a_1 \ldots a_m) = c \) and \( g(b_1 \ldots b_n) = d \) for some \( c, d \in M \) for which \( c \preceq d \), and this is clearly enough to establish the lemma, since \( \preceq^* \) extends \( \preceq \).

**Lemma 2** Where the indicated property holds for \( \preceq \) and all \( a, b, c \in M \), for all \( a, b, c \in M^* \) the same property holds for \( \preceq^* \):

- If \( a \preceq^* b \) and \( b \preceq^* c \) then \( a \preceq^* c \).
- If \( a \preceq^* b \) then \( a \oplus c \preceq^* b \oplus c \), where \( \oplus \) is \&, \( \circ \), \( \vee \) or \( + \).
- If \( a \preceq^* b \) then \( b \rightarrow c \preceq^* a \rightarrow c \).
- If \( b \preceq^* c \) then \( a \rightarrow b \preceq^* a \rightarrow c \).
- If \( a \preceq^* b \) then \( a \neg c \preceq^* b \neg c \).
- If \( b \preceq^* c \) then \( a \neg b \preceq^* a \neg c \).
- If \( a \preceq^* b \) then \( (\forall x)a \preceq^* (\forall x)b \).
- If \( a \preceq^* b \) then \( (\exists x)a \preceq^* (\exists x)b \).

**Proof** The same program was used to prove this lemma, using as input the conclusion of each of the properties listed. The output in each case is the list of possibilities for \( a_1 \ldots a_m, b_1 \ldots b_n \) for which \( f(a_1 \ldots a_m) \neq \bot \) and \( g(b_1 \ldots b_n) \neq T \). In each case, either these are s.t. \( f(a_1 \ldots a_m) \preceq g(b_1 \ldots b_n) \) or the values are not compatible with the antecedent of the property, and so can be discarded (for example, values which give an antecedent of \( c \preceq^* \bot \) for \( c \in M \)).

The cases considered in this lemma correspond in the obvious way to the various derivation rules of our systems, and so with the previous lemma we have established a general basis for the first half of our conservative extension results. In each of these we need now only demonstrate that \( v^*(A) \preceq^* v^*(B) \) for all the axiom
CHAPTER 2. A FORMULATION OF RELEVANCE

forms \( A \Rightarrow B \) introduced with the new vocabulary, and that the new derivation rules preserve this property.

For future reference, we also give a result which will prove useful in proving analogous conservative extension results for the logics generated by our rule systems:

**Theorem 10** For every entailment system \( \mathcal{E}_+ \) let \( \text{Id}^\alpha \) be the \&-conjunction of all instances of Identity (\( A \Rightarrow A \)) in its vocabulary. For quantified systems \( \mathcal{Q}_+ \) we let \( \text{Id}^\alpha \) be the \&-conjunction of all instances of \( \forall(A \Rightarrow A) \) in the appropriate vocabulary. Note that these will always exist, since \& and \( \rightarrow \) (and \( \forall \) for quantified systems) are part of the basic vocabulary. Let \( \text{Id}^\beta \) be the corresponding conjunction of instances of Identity (or their universally closed forms) formed when we add some new connective \( \otimes \) of \( L \) to this vocabulary, and let \( v^* \) be the top or bottom extension of the canonical valuation \( v^\alpha \) for \( \mathcal{E}_+ \) (or \( \mathcal{Q}_+ \)) relative to \( \otimes \). Then \( v^*(\text{Id}^\beta) = v(\text{Id}^\alpha) \).

**Proof** For any formula \( A \) in the expanded vocabulary, \( v^*(A) \) is either \( T \), \( \bot \) or \( e(B) \) for some \( B \) in the original vocabulary. But then \( v^*(A \Rightarrow A) \) is either \( T \) or \( v(B \Rightarrow B) \) (since \( T \Rightarrow T = \bot \Rightarrow \bot = T \)). And for any \( B \) in the original vocabulary, \( B \Rightarrow B \) is an instance of Identity in the expanded vocabulary, and so a conjunct of \( \text{Id}^\beta \), and \( v^*(B \Rightarrow B) = v(B \Rightarrow B) \). For quantified systems, when \( A \) is an element of the expanded language, \( v^*(\forall(A \Rightarrow A)) \) is either \( \forall T \) or \( v(\forall(B \Rightarrow B)) \) for some \( B \) in the original language; and for any \( B \) in the original language, \( \forall(B \Rightarrow B) \) is a conjunct of \( \text{Id}^\beta \), and \( v^*(\forall(B \Rightarrow B)) = v(\forall(B \Rightarrow B)) \). So \( v^*(\text{Id}^\beta) \) is equal to an expression of the form \( a_1 \& a_2 \& \ldots \) where every \( v(B \Rightarrow B) \) (or \( v(\forall(B \Rightarrow B)) \)) is some \( a_i \), and each \( a_i \) is either one of these identity elements or \( T \) (\( \forall \) for quantified systems). Since \( a \& a = a \& T = T \& a = a \) for all \( a \) and \( \forall T = T \), this in turn is equal to an expression of the form \( b_1 \& b_2 \& \ldots \) where the \( b_i \) are all the elements of the form \( v(B \Rightarrow B) \) (or \( v(\forall(B \Rightarrow B)) \)) for \( B \) in the original vocabulary, and this is just \( v(\text{Id}^\alpha) \).

**Theorem 11** \( \mathcal{E}_+ \) is a conservative extension of \( \mathcal{E}_+ \).

**Proof** By bottom extension, letting \( v^*(A \Rightarrow B) = \bot \). For each implicational
axiom $C \Rightarrow D$ (including $D$, $T$ and $E$), $v^*(C) = \bot$, and the Monotonicity rules for implication produce results which have this property also.

**Theorem 12** $E_+^{\alpha \rightarrow}$ and $E_+^{\alpha \rightarrow F}$ are conservative extensions of $E_+^{\alpha}$.

**Proof** By top extension, letting $v^*(A \rightarrow B) = T$. All of the permission axioms (including $F \otimes$) take valuations with $T$ on the right, and the permission Monotonicity rules produce results with the same property. Notice the way that the introduction rule $F \otimes$ here necessitates a top extension method.

**Theorem 13** Where $E \notin \alpha$, $E_+^{\alpha \circ}$ is a conservative extension of $E_+^{\alpha}$.

**Proof** Immediate from Th. 6.

**Theorem 14** $E_+^{\alpha +}$ is a conservative extension of $E_+^{\alpha \circ}$.

**Proof** By bottom extension, letting $v^*(A + B) = \bot$. All of the new axioms give a valuation with $\bot$ on the left. Any conclusion reached by an application of $M+$ will also give a valuation with $\bot$ on the left, and this is the only new rule.

**Theorem 15** Where $E \notin \alpha$, $E_+^{\alpha +}$ is a conservative extension of $E_+^{\alpha}$.

The proof of this is immediate from the two previous results.

**Theorem 16** $Q E_+^{\alpha}$ is a conservative extension of $E_+^{\alpha}$.

**Proof** Immediate from Th. 8.

**Theorem 17** $E_+^{\alpha}$ is a conservative extension of $E_+Z$, for every $\alpha$.

**Proof** Let $\alpha = \beta \cup \gamma$, where $\beta$ is either null, $\circ$, $+$, $\sim$, $\circ \sim$ or $+ \sim$, and $\gamma \subseteq \{D,T,E,F\}$. Trivially, $E_+^{\alpha Z}$ is the same as either $E_+^{\beta Z}$ or $E_+^{\beta F}$ (if $F \in \gamma$). The arguments of Ths. 12, 13 and 14 can be modified in straightforward ways to show that both of these are conservative extensions to $E_+Z$, and the result follows by Th. 11.
2.10 Properties of Theories and Logics

2.10.1 Definitions

We repeat here some previous definitions. Given an entailment system $\mathcal{E}^\alpha$, an $\mathcal{E}^\alpha$-theory is any set of sentences $T$ in the language of $\mathcal{E}^\alpha$ closed under the following:

- If $A \in T$ and $\vdash_{\mathcal{E}^\alpha} A \Rightarrow B$ then $B \in T$.
- If $A, B \in T$ then $A \& B \in T$.

An important class of theories are those which reify the entailments provable in the underlying rule system. These are the regular $\mathcal{E}^\alpha$ theories. For propositional systems, a regular $\mathcal{E}^\alpha$ theory is one which contains $A \Rightarrow B$ whenever $A \Rightarrow B$ is provable in the rule system; for quantified systems we strengthen this to $\forall (A \Rightarrow B)$ (by Th. 4, if $A \Rightarrow B$ is provable then so is $A\{s/x\} \Rightarrow B\{s/x\}$ for every term $s$, so we take it that this strengthening is reasonable). We define the $\mathcal{E}^\alpha$-logic, denoted by $\mathcal{L}^\alpha$ (or $Q\mathcal{L}^\alpha$ for quantified systems), to be the smallest regular theory (this will always be uniquely defined). $\mathcal{L}^\alpha$ will be said to be entailment normal iff the converse of the regularity condition holds for it; that is, if for every $A \Rightarrow B \in \mathcal{L}^\alpha$, $A \Rightarrow B$ is provable in $\mathcal{E}^\alpha$. In other words, $\mathcal{L}^\alpha$ is entailment normal if the notion of logical implication which it provides coincides with the notion of entailment provided by the underlying rule system. In general, we will show that only systems without Enthymeme deliver entailment-normal logics.

The next result states for the record some fairly trivial properties of theories which we will sometimes use without comment in what is to follow.

**Theorem 18** Let $T$ be a $\mathcal{E}^\alpha_\star$-theory.

- If $D \in \alpha$ then if $A, A \Rightarrow B \in T$ then $B \in T$.
- If $T \in \alpha$ then if $A \Rightarrow B, B \Rightarrow C \in T$ then $A \Rightarrow C \in T$.
- $A \& B \in T$ iff $A, B \in T$. 
CHAPTER 2. A FORMULATION OF RELEVANCE

- If $o \in \alpha$ or $+ \in \alpha$ then if $A, B \in T$ then $A \circ B \in T$.

- If $+ \in \alpha$ then if $A+B \in T$ then $A \vee B \in T$.

- If $F \in \alpha$ then if $A, B \in T$ then $A \sim B \in T$.

2.10.2 ‘Constructing’ Theories and Logics

It will be useful to give a slightly more constructive account of theories, specifically by separating out the effects of Entailment Transitivity $T \Rightarrow$. To this end, for any entailment system $E^\alpha_+$ we let $M E^\alpha_+$ be the formal system generated by the axioms of $E^\alpha_+$ together with all the derivation rules of $E^\alpha_+$ except $T \Rightarrow$, and define the property $\vdash M E^\alpha_+ A \Rightarrow B$ accordingly.

Theorem 19 $\vdash E^\alpha_+ A \Rightarrow B$ iff $A \Rightarrow B$ is an element of the closure of \{ $C \Rightarrow D \vdash M E^\alpha_+ C \Rightarrow D$ \} under $T \Rightarrow$.

Proof Right to left is immediate, since clearly $\vdash E^\alpha_+ A \Rightarrow B$ if $\vdash M E^\alpha_+ A \Rightarrow B$, and $E^\alpha_+$ is closed under $RT$. We can demonstrate the other direction by showing that the transitive closure of the set remains closed under all the other $E^\alpha_+$ inference rules, and this is straightforward.

Now we give our pseudo-constructive definition of $E^\alpha_+$ theoryhood. We do this in terms of an operation $C^\alpha$ on sets of sentences, given as follows:

- Let $X$ be a set of sentences.
- $C^\alpha_0(X) = X$
- $C^\alpha_{n+1}(X) = \{B : A \in C^\alpha_n(X) \text{ and } \vdash M E^\alpha_+ A \Rightarrow B\} \cup \{C \& D : C, D \in C^\alpha_n\}$
- $C^\alpha(X) = \bigcup_{n \in \omega} C^\alpha_n(X)$

Theorem 20 $T$ is an $E^\alpha_+$-theory iff there is some $X$ s.t. $T = C^\alpha(X)$.

Proof Clearly if $T$ is an $E^\alpha_+$ theory then $T = C^\alpha(T)$, since $T$ is closed under Adjunction and $M E^\alpha_+$ entailment. So left to right is immediate. To show right to
left, first note that $C^\alpha(X)$ is closed under Adjunction and $\mathcal{M}\mathcal{E}_+^\alpha$ entailment. It remains to show that it is closed under $T\Rightarrow$; so let $A\Rightarrow B$ and $B\Rightarrow C$ be provable in $\mathcal{M}\mathcal{E}_+^\alpha$, and assume that $A \in C^\alpha(X)$. Then for some $n$, $A \in C_n^\alpha(X)$, so $B \in C_{n+1}^\alpha(X)$ and so $C \in C_{n+2}^\alpha(X) \subseteq C^\alpha(X)$.

This definition has only a pseudo-constructive nature, since the set of entailments provable in $\mathcal{M}\mathcal{E}_+^\alpha$ is clearly infinite; however it will do for our purposes. The following theorem establishes that $C^\alpha(X)$ is a Tarski-style consequence operation:

**Theorem 21**

- $X \subseteq C^\alpha(X)$
  - If $X \subseteq Y$ then $C^\alpha(X) \subseteq C^\alpha(Y)$
  - $C^\alpha(C^\alpha(X)) = C^\alpha(X)$

**Theorem 22** $\mathcal{L}_+^\alpha = C_+^\alpha(\text{Id}_+^\alpha)$ (where $\text{Id}_+^\alpha$ is, as previously, the set of all instances of Identity in the vocabulary of $\mathcal{E}_+^\alpha$, or the conjunction of the universal closures of all these instances, for quantified systems).

**Proof** We first show that $X = C^\alpha(\text{Id}_+^\alpha)$ is a regular $\mathcal{E}_+^\alpha$ theory. So let $A\Rightarrow B$ be provable in $\mathcal{E}_+^\alpha$; then $A\rightarrow A\Rightarrow A\rightarrow B$ is provable in $\mathcal{E}_+^\alpha$ ($M\rightarrow_1$), and so $A\rightarrow B \in X$, since $A\rightarrow A$ is clearly in $X$. For our quantified systems, steps of $M\forall$ give $\forall(A \rightarrow A)\Rightarrow \forall(A \rightarrow B)$, and so $\forall(A \rightarrow B) \in X$, since clearly $\forall(A \rightarrow A)$ is. To show that $X$ is the smallest regular $\mathcal{E}_+^\alpha$ theory, it suffices to note that every $A\Rightarrow A$ is provable in $\mathcal{E}_+^\alpha$; for then $\text{Id}_+^\alpha$ is an element of every regular theory, in particular of the $\mathcal{E}_+^\alpha$ logic, and the result follows by Th. 21.

The next result establishes the general coherence of our reading of $A\Rightarrow B$ as an entailment: $A$ entails $B$ iff $B$ is in the smallest $\mathcal{E}_+^\alpha$-theory containing $A$ iff every $\mathcal{E}_+^\alpha$-theory which contains $A$ contains $B$ also.

**Theorem 23** $\vdash \mathcal{E}_+^\alpha A\Rightarrow B$ iff $B \in C^\alpha(\{A\})$.

**Proof** Left to right is immediate. Right to left follows by induction on the construction of $C^\alpha$. 
• If \( B \in C_0^\alpha(\{A\}) \) then \( A = B \), and \( A \Rightarrow A \) is provable in \( E_+^\alpha \).

• Assume that for all \( B \in C_n^\alpha(\{A\}) \), \( A \Rightarrow B \) is provable in \( E_n^\alpha \). Show that the same holds for \( C_{n+1}^\alpha(\{A\}) \).
  
  - If \( B \in C_n^\alpha(\{A\}) \) also then the result follows by the inductive hypothesis.
  
  - If \( B = B_1 \& B_2 \) for some \( B_1, B_2 \in C_n^\alpha(\{A\}) \) then \( A \Rightarrow B_1 \) and \( A \Rightarrow B_2 \) are both provable, by the inductive hypothesis, and the result follows by \( M \& \), \( I \& \) and \( T \Rightarrow \).
  
  - If \( C \Rightarrow B \) is provable in \( ME_+^\alpha \) and \( C \in C_n^\alpha(\{A\}) \) then by the inductive hypothesis \( A \Rightarrow C \) is provable and the result follows by \( T \Rightarrow \).

As a corollary, \( A \Rightarrow B \) is provable in \( E_+^\alpha \) iff \( B \in T \) whenever \( A \in T \) for every \( E_n^\alpha \) theory \( T \). Again, left to right is immediate; right to left follows using Th. 21: if \( B \in T \) whenever \( A \in T \) then clearly \( B \in C_n^\alpha(\{A\}) \). Another consequence will be important in what is to follow, so we state it as a theorem:

**Theorem 24** \( A \in L_+^\alpha \) iff \( Id_+^\alpha \Rightarrow A \) is provable in \( E_+^\alpha \).

This follows immediately, using Th. 22. As a corollary, for quantified systems we have \((\forall x)A \in L_+^\alpha \) whenever \( A \in L_+^\alpha \). For if \( A \in L_+^\alpha \), \( Id_+^\alpha \Rightarrow A \) is provable in \( E_+^\alpha \); by \( M \forall \), \((\forall x)Id_+^\alpha \Rightarrow (\forall x)A \) is provable also, but \( Id_+^\alpha \) contains no free variables (being a conjunction of universal closures), and so by \( IV \) and \( T \Rightarrow \), \( Id_+^\alpha \Rightarrow (\forall x)A \) is provable and this gives the result.

### 2.10.3 Entailment Normality of Non-Enthymeme Logics

To establish the relationship between the provability of \( A \Rightarrow B \) in an entailment system without Enthymeme and the membership of \( A \Rightarrow B \) in the corresponding logic we will use *metavaluations* (see [Mey74]). For each system \( E_+^\alpha \) we define a metavaluation \( T^\alpha \), which (for our purposes) is just a designated set of sentences of the language of \( E_+^\alpha \). We will show that \( T^\alpha = L_+^\alpha \) whenever \( E \not\in \alpha \), and the entailment normality of the non-enthymeme systems will follow from the conditions on the metavaluations, which are as follows:
• \( p \notin T^\alpha \) for all atoms \( p \).

• \( A \& B \in T^\alpha \) iff \( A, B \in T^\alpha \).

• Where \( \circ \) or \( + \) is in \( \alpha \), \( A \circ B \in T^\alpha \) if \( A, B \in T^\alpha \).

• \( A \lor B \in T^\alpha \) iff \( A \in T^\alpha \) or \( B \in T^\alpha \).

• \( A + B \notin T^\alpha \), for all \( A, B \) (even if \( + \) is in \( \alpha \)).

• If \( F \) (and therefore \( \rightarrow \) also) is in \( \alpha \), \( A \rightarrow B \in T^\alpha \) iff \( A, B \in T^\alpha \); otherwise \( A \rightarrow B \notin T^\alpha \) for all \( A, B \) (even where \( \rightarrow \) is in \( \alpha \)).

• For quantified systems, \((\forall x)A \in T^\alpha \) iff \( A\{s/x\} \in T^\alpha \) for all terms \( s \) in the language, and \((\exists x)A \in T^\alpha \) iff \( A\{s/x\} \in T^\alpha \) for some term \( s \).

• \( A \rightarrow B \in T^\alpha \) iff

\[
- \vdash_{\mathcal{E}^\alpha_+} A \rightarrow B \quad \text{and} \quad - A \notin T^\alpha \text{ or } B \in T^\alpha
\]

The next theorem establishes the entailment normality of systems without Ent-thymeme.

**Theorem 25 (Entailment Normality)** Where \( E \notin \alpha \), \( A \rightarrow B \in \mathcal{L}^\alpha_+ (Q\mathcal{L}^\alpha_+) \) iff \( A \rightarrow B \) is provable in \( \mathcal{E}^\alpha_+ (Q\mathcal{E}^\alpha_+) \).

The proof of this theorem will use the following lemmas:

**Lemma 3** Where \( E \notin \alpha \), if \( \vdash_{\mathcal{E}^\alpha_+} A \rightarrow B \) then \( A \rightarrow B \in T^\alpha \).

**Proof** We have to show that where \( A \rightarrow B \) is provable in the rule system, and \( A \in T^\alpha \), then \( B \in T^\alpha \) also. We use induction on the construction of \( \mathcal{E}^\alpha_+ \). First note that \( A \circ B \in T^\alpha \) iff \( A \& B \in T^\alpha \) iff \( A, B \in T^\alpha \), and that \( A + B \notin T^\alpha \) for any \( A, B \). In a non-\( F \) system, \( A \rightarrow B \notin T^\alpha \) for any \( A, B \); otherwise \( A \rightarrow B \in T^\alpha \) iff \( A \& B \in T^\alpha \). Finally, \( T^\alpha \) is closed under Modus Ponens: if \( A, A \rightarrow B \in T^\alpha \) then \( B \in T^\alpha \).
• Basis. If \( A \to B \) is an axiom of \( \mathcal{E}_+^{\alpha} \) then \( B \in T^\alpha \) whenever \( A \in T^\alpha \). For example, take \( S+: (A \to C) \circ (B \to C) \Rightarrow A + B \to C \). If \( (A \to B) \circ (A \to C) \in T^\alpha \) then \( A \to C, B \to C \in T^\alpha \). Then it must be the case that \( A \to C \) and \( B \to C \) are both derivable in the entailment system, and easy moves tell us that \( A + B \to C \) is also. And \( A + B \notin T^\alpha \) for all \( A, B \), and so \( A + B \to C \in T^\alpha \). We leave it to the conscientious reader to demonstrate that the statement holds for the rest of our axioms.

• Inductive step. The statement is true for any \( A \to B \) which follows from premisses for which is true by means of a \( \mathcal{E}_+^{\alpha} \) derivation rule. We illustrate with the permission Monotonicity principle \( M \sim \_1 \), for an \( F \)-level entailment system. Let the statement hold for some \( B \to C \) provable in the entailment system. Let \( A \sim B \in T^\alpha \). Then \( A \& B \in T^\alpha \), and so \( B \in T^\alpha \). By the inductive hypothesis, \( B \to C \in T^\alpha \), and so \( C \in T^\alpha \) by Modus Ponens. So \( A \& C \) and therefore \( A \sim C \) are both elements of \( T^\alpha \).

Lemma 4 Where \( E \notin \gamma, T^\alpha = \mathcal{L}_+^{\alpha} \).

Proof First note that \( T^\alpha \) is a regular \( \mathcal{E}_+^{\alpha} \) theory; it contains the implicational form of every \( \mathcal{E}_+^{\alpha} \) entailment (and the universal closure of each such form for quantified systems), and it is closed under Adjunction and Detachment (and therefore under \( \mathcal{E}_+^{\alpha} \) entailment also). So \( \mathcal{L}_+^{\alpha} \subseteq T^\alpha \). The other direction follows by induction on complexity of formulas.

• First note that whenever \( A \in T^\alpha \), \( A \) is compounded from elements \( B \to C \) using \( \&, \lor, \circ, \sim, (\forall x) \) and \( (\exists x) \). So the basis of the induction is the case where \( B \to C \in T^\alpha \); then \( B \to C \) is provable in \( \mathcal{E}_+^{\alpha} \) and hence \( B \to C \in \mathcal{L}_+^{\alpha} \).

• Assume that \( B \in \mathcal{L}_+^{\alpha} \) whenever \( B \in T^\alpha \) for all \( B \) shorter than \( A \), and let \( A \in T^\alpha \). If \( A = B \circ C \) (\( B \& C \), \( B \to C \)), then \( B, C \in T^\alpha \), and so \( B, C \in \mathcal{L}_+^{\alpha} \) by the inductive hypothesis, and \( A \in \mathcal{L}_+^{\alpha} \) follows by Th. 18. If \( A = B \lor C \), then \( B \in T^\alpha \) or \( C \in T^\alpha \), so \( B \in \mathcal{L}_+^{\alpha} \) or \( C \in \mathcal{L}_+^{\alpha} \), and the result follows by Th. 18 again. If \( A = (\forall x) B \) then \( B\{s/x\} \in T^\alpha \) for all \( s \) and so \( B\{s/x\} \in \mathcal{L}_+^{\alpha} \) for all \( s \) by the inductive hypothesis. But then \( B \in \mathcal{L}_+^{\alpha} \), and the result follows
by Th. If \( A = (\exists x)B \) then \( B\{s/x\} \in T^\alpha \) for some \( s \), and so by the inductive hypothesis \( B\{s/x\} \in L^\alpha_+ \) and the result follows by \( W3 \).

**Proof of Theorem** By Lemma 3, \( L^\alpha_+ = T^\alpha \), and by the metavaluation, \( A \Rightarrow B \) is provable in \( E^\alpha_+ \) whenever \( A \rightarrow B \in T^\alpha \). The other direction is of course given by the definition of a logic.

As a corollary we have:

**Theorem 26** Where \( E \not\in \alpha \), \( L^\alpha_+ \) is closed under Detachment; that is, where \( A, A \rightarrow B \in L^\alpha_+ \), \( B \in L^\alpha_+ \) also.

The proof is immediate: if \( A \rightarrow B \in L^\alpha_+ \) then \( A \Rightarrow B \) is provable in \( E^\alpha_+ \), and \( L^\alpha_+ \) is an \( E^\alpha_+ \) theory.

### 2.10.4 Entailment and Logical Implication in Enthymeme Systems

Systems with the enthymeme principle \( E \) will not in general be entailment normal. For example, \( A \rightarrow A \rightarrow (A \rightarrow A) \Rightarrow (B \rightarrow B) \in L^\alpha_+ \) but \( A \rightarrow A \rightarrow (A \rightarrow A) \Rightarrow (B \rightarrow B) \) is not provable in \( E^\alpha_+ \).

**Theorem 27** Where \( E \in \alpha \), \( A \rightarrow B \in L^\alpha_+ \) \((Q L^\alpha_+)\) iff either \( A \Rightarrow B \) or \( A \Rightarrow B \) is provable in \( E^\alpha_+ \) \((QE^\alpha_+)\).

We will again use metavaluations to establish the result. The only change we make to the non-enthymeme case is in the condition for implication, which becomes:

- \( A \rightarrow B \in T^\alpha \) iff
  - Either \( \vdash E^\alpha_+ A \Rightarrow B \) or \( \vdash E^\alpha_+ A \Rightarrow B \Rightarrow B \) and
  - Either \( A \not\in T^\alpha \) or \( B \in T^\alpha \).

**Lemma 5** \( T^\alpha \subseteq L^\alpha_+ \)
CHAPTER 2. A FORMULATION OF RELEVANCE

Proof The argument is as for Lemma 4, the only addition being that we need to show $A \rightarrow B \in \mathcal{L}_+^\alpha$ whenever $A \circ \text{Id}^\alpha \Rightarrow B$ is provable in the entailment system. If it is, then $A \circ \text{Id}^\alpha \rightarrow B$ is certainly in the logic, and of course $\text{Id}^\alpha$ is also. The result follows by simple moves and Enthymeme.

Lemma 6 $\mathcal{L}_+^\alpha \subseteq T^\alpha$

This will follow if we can show that $A \rightarrow B \in T^\alpha$ whenever $A \Rightarrow B$ is provable. For $T^\alpha$ is closed under Adjunction and Detachment, and so it will be a regular $\mathcal{E}_+^\alpha$-theory if it reifies $\mathcal{E}_+^\alpha$ entailment. The only addition to Lemma 5 needed is to consider the base case for the Enthymeme principle: $A \circ (A \circ B \circ C) \Rightarrow B \circ C$. We want to show that under the assumptions $A, A \circ B \circ C \in T^\alpha$, $B \circ C \in T^\alpha$ also.

Since $T^\alpha \subseteq \mathcal{L}_+^\alpha$ we know that $\text{Id}^\alpha \Rightarrow A$ is provable (Th. 24). Since $A \circ B \circ C \in T^\alpha$, either $A \circ B \Rightarrow C$ or $A \circ B \circ \text{Id}^\alpha \Rightarrow C$ is provable; in either case easy moves tell us that $B \circ \text{Id}^\alpha \Rightarrow C$ is provable also. This gives us the first half of what we need to show that $B \circ C \in T^\alpha$; the other half requires us to show that $C \in T^\alpha$ when $B \in T^\alpha$.

Under the assumptions, $C \in T^\alpha$ whenever $A \circ B \in T^\alpha$. But $A \in T^\alpha$, and so if $B \in T^\alpha$, $A \circ B \in T^\alpha$ too.

Proof of Theorem By the lemmas, $\mathcal{L}_+^\alpha = T^\alpha$, and by the metavaluation condition, either $A \Rightarrow B$ or $A \circ \text{Id}^\alpha \Rightarrow B$ is provable whenever $A \rightarrow B \in T^\alpha$. The other direction follows from the definition of logics in general, and by noting that $A \rightarrow B \in \mathcal{L}_+^\alpha$ whenever $A \circ \text{Id}^\alpha \Rightarrow B$ is provable (using simple moves and Enthymeme).

Again we have the corollary that Enthymeme logics are closed under detachment.

For the non-enthymeme logics, the fact that $\mathcal{E}_+^\alpha$-theories are closed under detachment with respect to $\mathcal{L}_+^\alpha$ logical implication follows trivially from entailment normality. For enthymeme systems only regular theories have this property in general:

Theorem 28 Where $E \in \alpha$ and $T$ is a regular $\mathcal{E}_+^\alpha$-theory and $A \rightarrow B \in \mathcal{L}_+^\alpha$, if $A \in T$ then $B \in T$ also.

Proof If $A \rightarrow B \in \mathcal{L}_+^\alpha$ then either $A \Rightarrow B$ is provable in the entailment system, in which case the result is immediate; or $A \circ \text{Id}^\alpha \Rightarrow B$ is provable, in which case the
result follows by noting that $Id^\alpha$ is an element of every regular theory.

For Enthymeme systems with Detachment as a basic principle we have the following:

**Theorem 29** $A \rightarrow B \in \mathcal{L}_+^\alpha$ iff $A \circ Id^\alpha \Rightarrow B$ is provable in $\mathcal{E}_+^\alpha$.

**Proof** Right to left is given by Th. 27. For the other direction, $Id^\alpha \Rightarrow A \rightarrow B$ is provable if $A \rightarrow B$ is in the logic; $A \circ Id^\alpha \Rightarrow A \circ (A \rightarrow B) \Rightarrow B$ follows by Monotonicity and Detachment.

### 2.10.5 Some Properties of Enthymeme Logics

The next theorem demonstrates the power of the Enthymeme principle, and together with the one following will provide a vital part in determining the expressive power of our implementation of a conditional logic programming language.

**Theorem 30 (Exportation)** Where $E \in \alpha$, $A \circ B \rightarrow C \rightarrow A \rightarrow B \rightarrow C \in \mathcal{L}^\alpha$.

**Proof** By Th. 27 it is enough to show that $(A \circ B \rightarrow C) \circ Id^\alpha \Rightarrow A \rightarrow B \rightarrow C$ is provable in $\mathcal{E}_+^\alpha$. We know that $Id^\alpha \Rightarrow (A \circ B \rightarrow C) \circ A \rightarrow B \rightarrow C$ is provable (by the $E$ principle, commutativity of $\circ$ and Th. 24), and by $M \circ$ and $E$ we have

$$(A \circ B \rightarrow C) \circ Id^\alpha \Rightarrow (A \circ B \rightarrow C) \circ ((A \circ B \rightarrow C) \circ A \rightarrow B \rightarrow C) \Rightarrow A \rightarrow B \rightarrow C$$

Some consequences of this important result:

**Suffixing** Where $T, E \in \alpha$, $A \rightarrow B \rightarrow B \rightarrow C \rightarrow A \rightarrow C \in \mathcal{L}^\alpha$ (By Exportation on the implicational counterpart of $To$.)

**Prefixing** Where $T, E \in \alpha$, $B \rightarrow C \rightarrow A \rightarrow B \rightarrow A \rightarrow C \in \mathcal{L}^\alpha$. (As for Suffixing except that we also use $Co$.)

**Assertion** Where $D, E \in \alpha$, $A \rightarrow A \rightarrow B \rightarrow B \in \mathcal{L}^\alpha$. (By Exportation on the implicational counterpart of $Do$.)
CHAPTER 2. A FORMULATION OF RELEVANCE

**Factor** Where \(D, E \in \alpha, A \rightarrow B \rightarrow A \odot C \rightarrow B \odot C \in \mathcal{L}^\alpha\). \((A \odot C)(A \rightarrow B) \Rightarrow B \odot C\) follows by principles for \(\odot\) and \(D \odot\), and the result by Exportation on the implicational counterpart of this.)

**Summation** Where \(E \in \alpha, A \rightarrow B \rightarrow A \odot C \rightarrow B \odot C \in \mathcal{L}^\alpha\). \(((A \odot C)(A \rightarrow B) \Rightarrow B \odot C\) follows by \(D \odot, D \odot\), Commutativity and Monotonicity, and the result by Exportation.)

**Bundling** Where \(D, E \in \alpha, A \rightarrow B \rightarrow A \odot B \in \mathcal{L}^\alpha\). (Exporting on \(A \odot B \rightarrow A \odot B\).)

Some points to note:

- Prefixing and Suffixing (and Assertion) here rely on Enthymeme. In this we agree with a strand running from Lewis in [LL32, p496] to Routley in [SMPB82, Ch.3] - the underlying ‘real’ implication is \((A \rightarrow B) \odot (B \rightarrow C) \rightarrow (A \rightarrow C)\). In contrast, many analyses of logical implication and entailment ([AB75] for example) take Prefixing and Suffixing as fundamental principles. There need not be a great dispute between these two positions, of course, since Enthymeme is no doubt a very natural principle for implication. It is nevertheless instructive to separate out the intuitions which lead to the disagreement. See [SMPB82, Ch.3] for a discussion of intuitive objections to Prefixing, Suffixing, Assertion and other commonly accepted principles as non-enthymematic expressions of content inclusion.

- We do not have Factor (Summation) principles for \& (V): these would allow us to derive \(A \rightarrow B \rightarrow A \& C \rightarrow C\), for example, which is clearly objectionable on relevant grounds.

- We neither have nor not want Exportation for \&: for then \(A \rightarrow B \rightarrow A\) would follow immediately from \(A \& B \rightarrow A\), and this is of course our main enemy.

The converse of Exportation requires Detachment as well:

**Theorem 31 (Importation)** Where \(D, E \in \alpha, (A \rightarrow B \rightarrow C) \rightarrow A \odot B \rightarrow C\) in \(\mathcal{L}^\alpha\).
CHAPTER 2. A FORMULATION OF RELEVANCE

Proof \( AoBo(A \rightarrow B \rightarrow C) \implies BoAo(A \rightarrow B \rightarrow C) \) (Monotonicity and Commutativity) \( \implies Bo(B \rightarrow C) \) (Monotonicity and Detachment) \( \implies C \) (Detachment). The result follows by Exportation on the implicational counterpart.

An important consequence of Importation is:

**Contraction** Where \( D, E \in \alpha, (A \rightarrow A \rightarrow B) \rightarrow A \rightarrow B \in \mathcal{L}^\alpha. \) (By Importation, \( (A \rightarrow A \rightarrow B) \rightarrow AoA \rightarrow B, \) and the result follows by \( Io \) and Monotonicity principles for \( \rightarrow. \))

### 2.10.6 Conservative Extensions

Here we carry over some of the conservative extension results for rule systems to the corresponding logics. A logic \( \mathcal{L}^\alpha \) is a conservative extension of some other logic \( \mathcal{L}^\beta \) if for every wff \( A \) in the common vocabulary of \( \mathcal{L}^\alpha \) and \( \mathcal{L}^\beta, A \) is an element of \( \mathcal{L}^\alpha \) iff it is an element of \( \mathcal{L}^\beta. \)

**Theorem 32** Where \( E \notin \alpha, \) and \( \beta \) is either \( \alpha0 \) or \( \alpha+, \mathcal{L}_+^\beta \) is a conservative extension of \( \mathcal{L}_+^\alpha. \)

We need to show that where \( A \in \mathcal{L}_+^\alpha \) but \( A \notin \mathcal{L}_+^\beta, A \) contains some occurrence(s) of \( 0 \) or \( +. \) The result follows straightforwardly from Th. 6 and Th. 24; if \( Id^\beta \Rightarrow A \) is provable in \( \mathcal{E}_+^\beta \) for \( A \) containing no \( 0 \) or \(+, \) then \( Id^\Rightarrow A \) is provable in \( \mathcal{E}_+^\alpha, \) where \( Id^\Rightarrow \) results from \( Id^\beta \) by replacing \( 0 \) and \( + \) by \& and \( \lor \) respectively. Clearly, \( Id^\Rightarrow \leftrightarrow Id^\Rightarrow \) is provable in \( \mathcal{E}_+^\alpha; \) hence \( Id^\Rightarrow A \) is provable, and so \( A \in \mathcal{L}_+^\alpha. \)

**Theorem 33** \( \mathcal{L}_+^\Rightarrow A \) and \( \mathcal{L}_+^{\Rightarrow F} \) are conservative extensions of \( \mathcal{L}_+^\alpha. \)

Proof Assume that \( Id^\Rightarrow A \) is not provable in \( \mathcal{E}_+^\alpha, \) where \( A \) is in the vocabulary of \( \mathcal{E}_+^\alpha. \) Then for the canonical valuation \( v^\Rightarrow \) in the formula algebra of \( \mathcal{E}_+^\alpha, v^\Rightarrow(Id^\Rightarrow) \neq v^\Rightarrow(A). \) But for the top-extension valuation \( v^\Rightarrow \) of Th. 12, \( v^\Rightarrow(A) = v(A) \) and \( v^\Rightarrow(Id^{\Rightarrow F}) = v(Id^\Rightarrow) \) (by Th. 10) = \( v^\Rightarrow(Id^\Rightarrow), \) and so \( v^\Rightarrow(Id^{\Rightarrow F}) \neq v^\Rightarrow(A), \) and \( Id^{\Rightarrow F} \Rightarrow A \) is not provable in \( \mathcal{E}_+^{\Rightarrow PF} \). (A similar argument establishes the case for \( \mathcal{L}_+^{PF} \).
CHAPTER 2. A FORMULATION OF RELEVANCE

Theorem 34 $\mathcal{L}_+^{\omega}$ is a conservative extension of $\mathcal{L}_+^{\omega}$.

The proof is similar to the previous theorem's, using Th. 14 instead of Th. 12.

Theorem 35 $\mathcal{Q}\mathcal{L}_+^{\omega}$ is a conservative extension of $\mathcal{L}_+^{\omega}$.

The proof is similar to that for Th. 32, using Th. 8 instead of Th. 6.

2.10.7 Comparison With Other Systems

Weak Relevant Logics and Non-Enthymeme Systems

In this section we compare some elements of our hierarchy with the positive fragments of the weak relevant logics $B$, $DK$ and $DA$. We present these results largely for the sake of completeness, and refer readers to [SMBP82, Ch. 4] for a detailed discussion of the properties and interest of the systems.

Take the following Hilbert-style axiomatization of the 'basic' positive relevant logic $B_+$ (cf [SMBP82, Sec. 4.1]):

Axioms

A1 $A \rightarrow A$

A2 $A \& B \rightarrow A$, $A \& B \rightarrow B$

A3 $A \rightarrow A \lor B$, $B \rightarrow A \lor B$

A4 $A \& (B \lor C) \rightarrow A \& B \lor A \& C$

A4 $(A \rightarrow B) \& (A \rightarrow C) \rightarrow A \rightarrow B \& C$

A5 $(A \rightarrow C) \& (B \rightarrow C) \rightarrow A \lor B \rightarrow C$

Rules

Adjunction $A, B \vdash A \& B$

Detachment $A, A \rightarrow B \vdash B$
CHAPTER 2. A FORMULATION OF RELEVANCE

Suffixing \( A \rightarrow B \vdash B \rightarrow C \rightarrow A \rightarrow C \)

Prefixing \( A \rightarrow B \vdash C \rightarrow A \rightarrow .C \rightarrow B \)

The rules, as [SMPB82, p.287] makes clear, are not intended as closure operations on theories, but rather just as machinery for generating new theorems of the logic. Thus Prefixing does not stipulate that \( C \rightarrow A \rightarrow .C \rightarrow B \) follows from \( A \rightarrow B \) in a sense which would require that any theory which contains the second must contain the first; rather, it states that where \( A \rightarrow B \) is a theorem of the logic, so is \( C \rightarrow A \rightarrow .C \rightarrow B \). These rules should be compared with the Monotonicity rules for implication in our formulation.

The logic \( DK_+ \) adds the axiom form of Transitivity:

\[(A \rightarrow) \& (B \rightarrow C) \rightarrow A \rightarrow C.\]

The logic \( DA_+ \) adds the axiom form of Detachment to \( DK_+ \):

\[A \& (A \rightarrow B) \rightarrow B.\]

Theorem 36

- \( L_+ = B_+ \).

- \( L^T_+ = DK_+ \).

- \( L^{DT}_+ = DA_+ \).

Proof To show that \( X \subseteq L^\alpha_+ \) (where \( X \) is \( B_+ \), \( DK_+ \) or \( DA_+ \), and \( \alpha \) is respectively \( \{} \), \( T \) or \( TD \)), we observe that each of the \( X \) axioms is in \( L^\alpha_+ \), and that \( L^\alpha_+ \) is closed under the \( X \) rules: we know \( L^\alpha_+ \) is closed under Adjunction (since it is a \( E\alpha_+ \) theory), and Detachment and the Affixing rules follow from the entailment normality of \( L^\alpha_+ \). For let \( A \rightarrow B \in L^\alpha_+ \). Then by entailment normality we know that \( A \rightarrow B \) is provable in \( E\alpha_+ \); hence by \( M \rightarrow 1 \), \( B \rightarrow C \rightarrow A \rightarrow C \) is provable also, and so \( B \rightarrow C \rightarrow .A \rightarrow C \in L^\alpha_+ \). The case for the other Monotonicity rule is similar. For the other direction, we need only show that \( A \rightarrow B \) is a theorem of \( X \) whenever \( A \rightarrow B \) is provable in \( E\alpha_+ \); for then we will have shown that it is a regular
\( \mathcal{E}_+^\alpha \) theory, since it is closed under entailment detachment and Adjunction. The proof is a straightforward induction on the construction of \( \mathcal{E}_+^\alpha \).

With Ths. 32, 33 and 34 note that we also have the following result:

**Theorem 37** When \( E \not\in \alpha \),

- \( \mathcal{L}_+^\alpha \) is a conservative extension of \( B_+ \).
- \( \mathcal{L}_+^{\alpha T} \) is a conservative extension of \( D K_+ \).
- \( \mathcal{L}_+^{\alpha DT} \) is a conservative extension of \( D A_+ \).

We can obtain the quantified versions of these logics, \( BQ_+ \), \( D K Q_+ \) and \( D A Q_+ \), by and adding the following:

**Axioms**

- **Q1** \((\forall x)A \rightarrow A\{s/x\}\) for all \( s \).
- **Q2** \(A\{s/x\} \rightarrow (\exists x)A\) for all \( s \).
- **Q3** \((\forall x)(A \rightarrow B) \rightarrow .A \rightarrow (\forall x)B\).
- **Q4** \((\forall x)(B \rightarrow A) \rightarrow .(\exists x)B \rightarrow A\).
- **Q5** \(A \& (\exists x)B \rightarrow .(\exists x)(A \& B)\).
- **Q6** \((\forall x)(A \lor B) \rightarrow .A \lor (\forall x)B\).

**Rules**

**Universalization** \( A \vdash (\forall x)A \)

We have the usual restriction, that \( x \) must not occur free in \( A \), in Q3 to Q6. Note that the Universalization rule is intended in the same sense that the propositional rules had; if \( A \) is a theorem then so is \( (\forall x)A \).

**Theorem 38** \( \bullet Q\mathcal{L}_+ = BQ_+ \)
Proof To show that the Hilbert formulations are contained in our systems we need only note that each of Q1 to Q6 are elements of $\mathcal{Q}^T_+$ (and hence also of the other two logics), and refer to Th. 24 for the closure of $\mathcal{Q}^T_+$ under the Universalization rule. For the other direction we need to show that each Hilbert formulation is a regular theory of the appropriate type, which amounts to showing that it contains $\forall (A\rightarrow B)$ whenever $A\Rightarrow B$ is provable in the corresponding entailment system. The proof proceeds by the normal induction on the construction of the entailment systems; each of the quantifier axioms has the obvious counterpart in Q1 to Q6, and that $MV$ and $M3$ preserve the property is demonstrated by the following argument: Let $A\rightarrow B$ be a theorem of the Hilbert formulation. Then $(\forall x)A\rightarrow B$ is also, using Q1, Prefixing and Detachment. By Universalization, $(\forall x)((\forall x)A\rightarrow B)$ is a theorem, and since $x$ does not occur free in $(\forall x)A$, $(\forall x)A\rightarrow (\forall x)B$ follows by Q3 and Detachment. This establishes the argument for $MV$ and $M3$ yields to a similar treatment.

‘$R_+$ is an Enthymematic Extension to $DA_+$’

$R$ is generally recognized as the major relevant logic, and here we will show that it can be characterized as an Enthymeme system in our hierarchy. Whether or not readers agree with us in finding our formulation a much more gentle and illuminating pathway to $R$ than those which exist in the literature, it has at least some technical interest in making precise the intuitions, summarized in [SMPB82], that many of the principles taken as fundamental in $R$ are only enthymematically true.

Take the following axiomatization of $R_+$ (with fusion primitive - cf [AB75, §27]):

Axioms

A1 $A\rightarrow A$
CHAPTER 2. A FORMULATION OF RELEVANCE

A2  \( A \rightarrow B \rightarrow .B \rightarrow C \rightarrow .A \rightarrow C \)

A3  \( A \rightarrow .A \rightarrow B \rightarrow B \)

A4  \( (A \rightarrow .A \rightarrow B) \rightarrow .A \rightarrow B \)

A5  \( A \& B \rightarrow A, A \& B \rightarrow B \)

A6  \( (A \rightarrow B) \& (A \rightarrow C) \rightarrow .A \rightarrow B \& C \)

A7  \( A \rightarrow A \vee B, B \rightarrow A \vee B \)

A8  \( (A \rightarrow C) \& (B \rightarrow C) \rightarrow .A \vee B \rightarrow C \)

A9  \( A \& (B \vee C) \rightarrow A \& B \vee C \)

A10 \( A \rightarrow .B \rightarrow A \& B \)

A11 \( (A \rightarrow .B \rightarrow C) \rightarrow A \& B \rightarrow C \)

Rules Detachment, Adjunction.

Theorem 39 \( L^{ODTE} = R_. \)

Proof Clearly A1, A5, A6, A7, A8 and A9 are elements of all our logics. We demonstrated previously that A2, A3, A4, A10 and A11 are provable in \( L^{ODTE} \), which is also closed under Adjunction and Detachment. So \( R_+ \) is contained in \( L^{ODTE} \). For the other direction, it is enough to show that \( R_+ \) reifies \( E^{ODTE} \) entailment; since it is closed under Adjunction and Detachment and contains every instance of Identity it will then be a regular \( E^{ODTE} \)-theory. This follows by a tedious but straightforward induction on the construction of \( E^{ODTE} \).

Since \( L^{ODT} \) is a conservative extension of \( DA'_+ \) (Th. 37) this result establishes our characterization of \( R_+ \) as an enthymematic extension to (a conservative extension with \( \circ \) of) \( DA'_+ \). One of the problems with the Hilbert style axiomatization given here is that it contains nested implications, objects which lack a well-established place in our intuitions. Another is the difficulty of gleaning from the axioms much feeling for the meaning of fusion. Fusion has always had a slightly uneasy position in relevant logics; formally it is extremely useful and well-behaved, but there is no
very firm commitment to an intuitive sense for it (see [AB75, p345]). One of the things we like about our construction is that it deals with both of these problems at once; we formulate $R_+$ without any significant appeal to nested implicational principles, relying instead on simple properties of fusion which here has a clearly defined sense as an intensional conjunction, distinguished from its extensional counterpart by not supporting Weakening $(A_0B \Rightarrow A)$. In Chapter 4 we show that fusion plays a vital role in allowing us to express a computationally viable fragment of $R$.

With Th. 33 and Th. 34 note that we also have the following result:

**Theorem 40** $\mathcal{L}^\alpha_{\text{DTE}}$ is a conservative extension of $R_+$, where $\alpha$ is one of $0\to$, $0\to F$, $+, +\to$ and $+\to F$.

If we add the quantifier axioms Q1 to Q6 and Universalization to $R_+$ we have the logic $RQ_+$.

**Theorem 41** $Q\mathcal{L}^\alpha_{\text{DTE}} = RQ_+.$

**Proof** The only addition to the arguments of Th. 38 and Th. 39 concerns the axiom $Do\exists$: we leave it to the reader to show that $A_0(\exists x)B \Rightarrow (\exists x)(A_0B)$ (for $x$ not free in $A$) is a theorem of $RQ_+$.

### 2.11 Logic-Based Theories

As we discussed briefly in Sec. 2.1, the more standard definition of a theory in relevant circles is based on closure under logical implication for some given logic (see [Mey74]). For a given logic $\mathcal{L}$, a $\mathcal{L}$-theory is any set of sentences closed under:

If $A \in T$ and $A \Rightarrow B$ is a theorem of $\mathcal{L}$ then $B \in T$.

Generally, we would continue to be interested only in the theories which are closed under Adjunction as well. A regular $\mathcal{L}$-theory is one which contains all theorems of $\mathcal{L}$. For non-enthymematic systems the two notions coincide, as a consequence.
CHAPTER 2. A FORMULATION OF RELEVANCE

of the entailment-normality of these systems. By the same token, this is not the case for enthymematic systems, which are not entailment normal. What we do have, however, is the following:

**Theorem 42** Let $E \in \alpha$. Then an $\mathcal{L}_\alpha$ theory is any set of sentences $T$ in the language of $\mathcal{E}_\alpha$ closed under Adjunction and the following:

If $A \in T$ and $AoId_\alpha \Rightarrow B$ or $A \Rightarrow B$ is provable in $\mathcal{E}_\alpha$ then $B \in T$.

This is a straightforward consequence of Th. 27. For systems which include Detachment as well, Th. 29 gives the following simplified condition:

If $A \in T$ and $AoId_\alpha \Rightarrow B$ is provable in $\mathcal{E}_\alpha$ then $B \in T$.

The two notions do however coincide on their definitions of regular theoryhood. Let $\mathcal{E}_+^\alpha$ be an Enthy1neme systen1 which generates a logic $\mathcal{L}_+^\alpha$. By definition, $\mathcal{L}_+^\alpha$ is contained in every regular $\mathcal{E}_+^\alpha$ theory, and Th. tells us that these are closed under $\mathcal{L}_+^\alpha$ logical implication. Conversely, every $\mathcal{L}$-theory will be an $\mathcal{E}$-theory for an entailment system $\mathcal{E}$ which generates $\mathcal{L}$, since $A \Rightarrow B$ is an element of the logic whenever $A \Rightarrow B$ is provable in the entailment system (by definition). So the differences are perhaps not important, if we are only interested in regular theories.

On the other hand, we believe that the non-regular theories are important in their own right, to the extent that they allow us to model the kind of productive implication discussed in Sec. 2.1. To return to our planning example of the first chapter, we can imagine situations where it might be at least annoying to be told that having a factory in zone2 is a consequence of having a factory in zone2. And the loop detection example we sketched in that chapter simply will not work if we interpret our database as a regular theory.

The logic-based definition of theories has an important property (for strong Enthymeme logics): $A \Rightarrow B \Rightarrow C \in T$ iff $AoB \Rightarrow C$ (by Exportation and Importation). As we will see in Chapter 4, this property is crucial in determining the expressive power of the fragment of the logic we can compute by a simple extension
CHAPTER 2. A FORMULATION OF RELEVANCE

to the SLD (and standard PROLOG) model. Unfortunately, the entailment system definition of theoryhood does not share this property for any of the systems discussed in this chapter. Because we know of no efficient way to compute non-regular theories in this case, pragmatic considerations will force us to interpret our databases as theories according to the logic-based scheme; we are unhappy with this situation, and we will attempt to show why in the remainder of this section.

Here will give a formulation of $R_+$ using a new entailment system $\mathcal{E}_+^R$ which includes the Residuation principle. This differs in no particularly important way from the algebraic development of [Dun66] and [MR72] (which we will only briefly discuss here). $R_+$ will be entailment normal with respect to $\mathcal{E}_+^R$, so the desirable property noted above will hold of $\mathcal{E}_+^R$ theories. The principles we take for $\mathcal{E}_+^R$ include all the zero-degree principles of $\mathcal{E}_+^O$, together with $Do$ and

**Residuation** $AoB \Rightarrow C \vdash B \Rightarrow A \rightarrow C$

We leave it to the reader to show that all other implicational principles (that is, $To$, $S&$, $SV$, $So$, $M \rightarrow_1$, $M \rightarrow_2$ and $E$) are derivable in this system.

The logic generated (in the normal way for our development) by these principles way is in fact $R_+$, but $R_+$ is not entailment normal with respect to the entailment system containing just these principles. In particular, let $B$ be a theorem of $R_+$; then so is $A \rightarrow AoB$, for any $A$ (using Exportation and Detachment on $AoB \rightarrow AoB$), but $A \Rightarrow AoB$ is not in general derivable. To get entailment normality we need one more principle:

**Truth Addition** $A \Rightarrow AoId^R$

where $Id^R$ is of course the conjunction of all instances of Identity in the language of $\mathcal{E}_+^R$.

Truth Addition is the root of our suspicion of logic-based theories; we simply fail to see any intuitive justification for it. Of course, it would follow if we had accepted the doctrine that necessary truths have 'no content' - so would $A \Rightarrow A & Id^R$,
leading quickly to irrelevance. In the algebraic development, $Id^R$ becomes the propositional constant $t$ ('truth'), which is an identity for $o$ in the algebra (analogously, we can derive $A \circ Id^R \leftarrow \rightarrow A$ in $E^R_+$), but while this might be a nice algebraic property it seems less convincing in non-mathematical interpretations.

To give an example of possible problems with the logic-based definition of theories (and hence of Truth Addition), note that $(A \rightarrow A) \rightarrow B \rightarrow B$ is a theorem of $R_+$ (using Exportation and Detachment on $(A \rightarrow A) \circ ((A \rightarrow A) \rightarrow B) \rightarrow B$), and so any theory (regular or not) which contains $A \rightarrow A \rightarrow B$ will also contain $B$, even if it does not contain $A \rightarrow A$. While this apparent oddity can be glossed over in various ways, it does serve to vitiate whatever practical applications we might have for non-regular theories, since we cannot exclude the effects of necessary truths to the extent that we might like. As a simple example, defining a predicate $\text{loop-on-f}$ as

$$(f \rightarrow f) \rightarrow \text{loop-on-f}$$

will not work; $\text{loop-on-f}$ will always be true.

In summary, then, our current knowledge of computational techniques is inadequate to handle non-regular theories based on non-residuating entailment systems, and so we are reluctantly forced to the formally more tractable but (to our mind) intuitively inferior logic-based definition.
Chapter 3

Negation

3.1 Setting the Agenda

3.1.1 'Positive' and 'Failure' Negations

Vanilla PROLOG does not allow negation in assertion positions; things of the form \( \neg A \) or \( B \rightarrow \neg A \) cannot be asserted. Negation can only occur in queries or in rule bodies, and is interpreted as failure: the query \( \neg A \) succeeds just in case the query \( A \) (finitely) fails. With these restrictions it is impossible for both of the queries \( A \) and \( \neg A \) to succeed. In our systems we want to introduce a different kind of negation, one not subject to these syntactic restrictions. It should be viewed as an extension to the positive systems, rather than as a replacement for NAF ('negation-as-failure'); we will briefly discuss the difficulties of adding NAF to our negated systems in the next Chapter, and though we will not in fact be implementing NAF directly, there is no objection to it in principle. In other words, we can envisage a two negation system; the first is one which allows negated assertions, a kind of 'positive' negation, and the other negation as failure in whatever form it must take to remain compatible with our extensions to the underlying positive language. We will also show (in Chapter 5) how NAF can to a certain extent be mimicked by our 'positive' negation.
3.1.2 Inconsistency and Paraconsistency

Allowing negated assertions introduces the possibility of explicitly inconsistent databases, a problem avoided by PROLOG. Our view of negation is decidedly non-classical, in that while an inconsistent database is taken to be undesirable, it is by no means catastrophic. A simple-minded classical reading of negation would require everything to be deducible from an inconsistent database. Of course, this is not what happens in practice; generally in AI systems we would incorporate some machinery for avoiding anomalies like inconsistencies. For example, we might see an inconsistency as generating a split in the database into its acceptable segments in the style of deKleer's ATMS ('Assumption Based Truth Maintenance') model ([deK86]). We will discuss consistency maintenance strategies in Chapter 5; for the moment it is sufficient to note that all such strategies will fail to be general in important ways (essentially because there is no effective method for determining the consistency of a set of sentences), and that they will always be expensive computationally. The major advantage of adopting a paraconsistent logic, one which does not take every B to be a consequence of an inconsistency \( A \& \neg A \), is that it provides a formal framework within which these expensive and incomplete strategies are theoretically optional.

To illustrate what we mean by this last statement, assume that the database contains some implicit contradiction on \( r \); say that \( r \) itself is in the database explicitly, and \( \neg r \) is derivable by a devious and highly-nonapparent chain of reasoning - perhaps our query evaluation procedure goes into a loop when presented with the goal \( \neg r \), even though it is in fact derivable from the database according to the background logic. Assume further that our system has an (incomplete) inconsistency detection mechanism, which this deeply buried contradiction has eluded. Now consider two queries \( p \) and \( q \), where the information in the database which is 'relevant' to \( p \) (in an intuitive, syntactic sense) is not enough to make \( p \) true, and where on the other hand \( q \) is derivable, but only in a way which uses \( r \). The answers we could expect to the two queries are 'No' and 'Yes' respectively. These answers are incomplete if our background logic is not paraconsistent, and we continue to take the database as a homogeneous whole: for of course then
the answer to \( p \) (and everything else) should be ‘Yes’. If on the other hand we see a contradiction as generating a split in the database then the answer of an unqualified ‘Yes’ to the query \( q \) is incorrect: the right answer is ‘Yes - in the following contexts which retain \( r \) but somehow reject \( \neg r: \ldots \)’. With respect to a paraconsistent logic, however, ‘No’ and ‘Yes’ will be the correct answers for the database taken as a whole. For the first answer, the effects of the inconsistency are not allowed to spread beyond its immediate ‘neighbourhood’, and so the status of \( r \) here has no logical connection with that of \( p \). For the second, the logic allows \( r \) and \( \neg r \) to co-exist and so there is no logical reason not to use \( r \) in the evaluation of \( q \).

To put the point another way, a standard logic will require us to find and deal with all inconsistencies, if we want to be sure of remaining formally coherent with respect to the logic. With a paraconsistent logic, we can view consistency maintenance strategies as pragmatic devices for removing some syntactically-detectable anomalies from our databases; if we do not find all of them, we do not thereby move outside the scope of situations which the logic is suited to. To us, the best paraconsistent viewpoint on contradictions is one which sees them as essentially no worse than any other falsehood;¹ the only thing which makes a theory which contains a false piece of information like ‘My name is David’ essentially different from one which contains two explicitly inconsistent facts like ‘My name is David’ and ‘My name is not David’, is that we know the latter must always be false, whereas we establish the falsity of the first empirically. But they are just as bad as each other; both contain false information, the only difference is that it may be easier to detect it in the second theory than it is in the first.

For this reason we believe that consistency maintenance is rightly viewed as a mechanism placed on top of an inconsistency-tolerating substrate, a device for removing some syntactically-detectable falsehoods from our databases. Without consistency maintenance, we get a more efficient computation; and although some of our answers may rely on information which is in fact false, this is essentially no different from the situation with consistency maintenance turned on; for we

¹However, there are those who would say that some contradictions are actually true - see the arguments in [PR84], for example
may be sure that any real-world database of significant size will contain false information, even if it is not inconsistent in the sense that both $A$ and $\neg A$ can be derived from it. The most we ensure with consistency maintenance is fewer falsehoods, at substantial computational cost.

For technical discussions of paraconsistency in general we refer readers to [Arr80], [PR84], [PR84] and [SMPB82].

The first application of paraconsistent logics (that we know of) in the general area of deductive databases and and logic programming was due to Belnap (in [Bel77a] and [Bel77b], applying some of Dunn’s ideas from [Dun76]), who proposed computer reasoning systems based on the first-degree implicational fragment $R_{fde}$ of $R$. In [Bol85] we developed an implementation based on this proposal (restricted mainly to the propositional level), with the difference that the underlying logic did not require Contraposition for implication (so that asserting a rule $A \rightarrow B$ to the system did not commit us to $\neg B \rightarrow \neg A$ as well). Although the structure of theories was different to those based on $R_{fde}$, the class of logical theorems was exactly the same, so we could argue that our weaker system was a coherent expression of the intuitions underlying $R_{fde}$. Our arguments against Contraposition were less well-developed versions of those that we give later in the chapter. Rejecting Contraposition allowed us to view atoms $p$ and $\neg p$ as completely independent, with the result that adding negation to the positive base required no new computational apparatus. For example, if we take the part of standard PROLOG without NAF, and extend its language in a trivial way by allowing $\neg p$ to be an atom (with no logical connection to the atom $p$) then the resulting system will compute the appropriate fragment (with the standard PROLOG restrictions on what may occur in assertion positions) of the quantified version of the logic we derived from $R_{fde}$ in [Bol85]. This follows from the fact that $R_{+fde}$ is identical to the first-degree implicational fragment of classical (and intuitionistic) logic, which is of course one description of the logical basis of PROLOG. Blair and Subrahmanian give a description of such a system in more standard logic programming terms (with model-theoretic and fix-point semantics) in [BS87].

\[2\] The claim that their negation differs from Belnap’s in the footnote on p343 of this paper would seem to result from the misprinted table for negation in [Bel77b] (corrected in [Bel77a]).
In this chapter we extend the nice property of being able to strongly represent the full logic within its positive fragment to systems of general degree.

Further arguments for the use of paraconsistent logics in Artificial Intelligence may be found in [Per86], [Fit85] and [Bat86]

### 3.1.3 Truth and Falsity

Before we add negation to our systems, we must settle the question of how we are intuitively to read $\neg A$. The answer is unsurprising:

- $\neg A$ is true iff $A$ is false.

The dual makes sense too:

- $\neg A$ is false iff $A$ is true.

The negation principles to be adopted for our systems will then depend intimately on what we believe to be the relationship between truth and falsity. We have no trouble accepting the following consistency and completeness requirements:

- Either $A$ is true or it is not true.
- Either $A$ is false or it is not false.
- $A$ is never both true and not true.
- $A$ is never both false and not false.

However we will abandon the traditional knitting up of truth and falsity:

- $A$ is true iff $A$ is not false.

Contraposition for implications ('generalized Horn clauses') is also rejected (though without any discussion) and it is straightforward to show that their system is isomorphic in its propositional part to that described in [Bol85] when the PROLOG restrictions on assertions are applied to the latter.
CHAPTER 3. NEGATION

- $A$ is false iff $A$ is not true.

which would give us:

- $A$ is always either true or false.
- $A$ is never both true and false.

To see why the second of these must be rejected, given other principles which we accept, refer back to the Lewis argument of Sec 1.6, which would force every inconsistent theory to be trivial in the sense of containing every sentence in the language.

In our systems, then, truth and falsity are not to be taken as exclusive; statements can be both true and false, which is just the situation when we assert both $A$ and $\neg A$ into a database. We can, if we like, interpret this as 'told true' and 'told false' (cf [Bel77a],[Bel77b]), to illustrate that there is no suggestion here that in the actual world things may be both true and false. The same is not true, however, for models of the world, and if such a model contains the hypothesis that $A$ is both true and false, we have no justification for importing into consideration of what holds in that model our understanding that the world is not really like that. We are certainly justified in concluding that the model fails to accurately reflect the real situation, but this is not to say that the model is totally without value; no map is totally accurate.

There is another Lewis argument, which purports to establish $B \lor \neg B$ follows from an arbitrary sentence $A$ ([LL32, p.258]):

- Assume $A$.
- $B$ is always either true or false, so given $A$, we have $A \& (B \lor \neg B)$.
- $B \lor \neg B$ follows from $A \& (B \lor \neg B)$.

Clearly, this argument will not convince us that $A \Rightarrow B \lor \neg B$ holds non-elliptically: we have imported an additional element, namely $B \lor \neg B$ itself, into our consideration of what follows from $A$. Remember that our entailment relation is always
non-elliptic; even if we take \( B \lor \neg B \) to be a necessary truth we are not allowed to import it without explicit mention into considerations of what follows from \( A \).

So far, our notion of necessary truth has been dependent entirely on our notion of entailment; the set of logical truths is just the theory which contains \( A \rightarrow B \) whenever \( A \) entails \( B \). There is no objection in principle to expanding this set to include every instance of Excluded Middle (ie, \( A \lor \neg A \)):

**Definition 3** A completely regular \( \mathcal{E}^\alpha \) theory is a regular \( \mathcal{E}^\alpha \) theory which includes every instance of \( A \lor \neg A \) (where ++ \( \epsilon \alpha \)) or \( A + \neg A \) (where ++ \( \epsilon \alpha \)).

This extended type of regular theory will in turn lead to an extended type of logic:

**Definition 4** The complete \( \mathcal{E}^\alpha \)-logic, denoted by \( \mathcal{L}^\alpha \), is the smallest completely regular \( \mathcal{E}^\alpha \) theory.

Complete logics and completely regular theories will not figure largely in our discussion (but see Sec. 3.8). The view we adopt here is that logic has mainly to do with \( \forall \) (hat follows from what, so that logical truths are fundamentally implicational.

Thus far we have given only a negative characterisation of the relationship between truth and falsity in our systems; it is neither exclusive nor exhaustive. In fact, we want to go further than this, and say that truth and falsity are essentially independent. At the most basic level, for example, we want to say that the atoms \( p \) and \( \neg p \) are essentially just different atoms, as we discussed above. This kind of independence will be carried up the scale of formula complexity; there will be no essential connection between the properties of \( A \rightarrow B \) and \( \neg(A \rightarrow B) \), for example. This mantra - 'Truth and falsity are independent' - will provide a simple but powerful guide for formulating negation principles.

We are not suggesting that truth and falsity in the real world are independent; indeed we believe that they probably are just exclusive and exhaustive. We have explained why an assumption of exclusiveness cannot be imported into the
CHAPTER 3. NEGATION

Maps of the world we will construct in our systems; such maps will in general be inconsistent to some extent, and an assumption of exclusiveness would be disastrous (and inappropriate). On the other hand, there would seem to be no major problem with including 'A or not A' in each of these maps, and we are perhaps open to charges of incompleteness for not doing so. However, it will always be the case that any real database will be at best an incomplete model of the world; even if we were to include Excluded Middle there would be an infinite number of gaps in our picture, and presumably most of these would be more regrettable than the lack of 'A or not A' (eg, true information about how to cure cancer, prevent nuclear war, live for ever, where I left my keys this morning etc. etc.).

There is in fact another agenda here. The principles we reject, most notably Excluded Middle and Contraposition for entailments:

\[ A \Rightarrow B \vdash \neg B \Rightarrow \neg A \]

are not well-behaved from a computational point of view, at least for the scheme used in our system. Rejecting these, we are able construct full systems which can be represented within their positive fragments, and so an evaluation procedure which is adequate in the positive case will extend, with only the most trivial modifications, to the full system. We should make it clear that we are not opposed to the rejected forms (or to the use of an exclusive and exhaustive reading of negation) as acceptable principles for everyday reasoning; rather, our position is that there is a stable, coherent position to the effect that they are not fundamental, but rely on hidden assumptions (which in 'everyday' situations are quite acceptable). While we are perhaps open to charges of pedantry in rejecting the principles, as a last resort we can at least claim to have given a coherent description of the fragment we are able to implement efficiently. See Sec. 3.7 for further discussion of the issues.

---

3As in Priest’s Logic of Paradox ([Pri79]), for example
3.1.4 Strategy for Formulating Negation Principles

With a traditional view of truth and falsity, principles for negation will come very easily, essentially as parasites on truth conditions. For when our systems tell us that

\[ A \Rightarrow B \]

is a valid entailment, we can always take it as natural to conclude

\[ \neg B \Rightarrow \neg A \]

If \( B \) is false, then \( B \) is not true, so \( A \) cannot be true, so \( A \) must be false. We will also have the principle of Double Negation:

\[ A \iff \neg \neg A. \]

If \( A \) is true, then \( A \) is not false, hence \( \neg A \) is not true, hence \( \neg A \) is false, hence \( \neg \neg A \) is true; and conversely. Together these principles will generally suffice to give most of the commonly accepted principles for negation.

With our reading of truth and falsity as independent, however, we are committed to finding independent falsity conditions; principles justified in their own right. That is, to find conditions for \( \neg A \) we must find natural falsity conditions for \( A \), given that we take it that \( \neg A \) is true iff \( A \) is false.

3.2 Falsity Conditions for ‘And’ and ‘Or’

We take the following falsity conditions as natural:

- \( A \& B \) is false iff \( A \) is false or \( B \) is false.
- \( A \lor B \) is false iff \( A \) is false and \( B \) is false.
- \( \neg A \) is false iff \( A \) is true.
CHAPTER 3. NEGATION

These lead straightforwardly to our first negation principles:

Negation

\[
N\& (A&B) \iff \neg A \lor \neg B
\]

\[
N\lor (A \lor B) \iff \neg A \land \neg B
\]

\[
N\neg \neg A \iff A
\]

They follow naturally from the falsity conditions, via the condition we took as basic for \( \neg A \); that is,

\( \neg A \) is true iff \( A \) is false.

To take \( \neg \neg \), for example: \( \neg \neg A \) is true iff \( \neg A \) is false iff \( A \) is true. These principles are completely in accord with the traditional principles (being just DeMorgan’s Laws and Double Negation ); the only difference is in the interpretation of truth and falsity as independent, rather than exclusive and exhaustive. The quantifier principles are equally traditional, and are just generalized forms of \( N\& \) and \( N\lor \):

\[
N\forall (\forall x)A \iff (\exists x)\neg A
\]

\[
N\exists (\exists x)A \iff (\forall x)\neg A
\]

The principles for fusion and fission follow by analogy with those for the extensional connectives:

\[
N\circ (A \circ B) \iff \neg A \lor \neg B
\]

\[
N+ (A+B) \iff \neg A \circ \neg B
\]

3.3 Negation and Implication

The question is: When can we conclude that \( A \rightarrow B \) is false? When is it false, that were \( A \) true then \( B \) would therefore be true? We think there are two possible answers:
- When \( B \) might be false even if \( A \) is true.

- When \( B \) might not be true even if \( A \) is true.

Given exclusive and exhaustive truth and falsity these answers are of course equivalent (since then failing to be true is exactly the same as being false).

We prefer the first, because the second can never hold if the truth of \( A \) is sufficient for the truth of \( B \) (ie, if \( A \rightarrow B \) is true), and so forces an exclusivity which is not in the spirit of this investigation. To put it another way, we take it that the second answer will give us a condition under which \( A \rightarrow B \) is not true, which for us is not the same thing as being false. Our falsity condition for \( A \rightarrow B \) will be:

\[
A \rightarrow B \text{ is false iff } B \text{ might be false even if } A \text{ is true iff the truth of } A \text{ permits the falsity of } B.
\]

which we formalize as

\[
N \rightarrow \neg(A \rightarrow B) \iff A \rightarrow \neg B
\]

For completeness, we have the dual

\[
N \rightarrow \neg(A \rightarrow B) \iff A \rightarrow \neg B
\]

That is, ‘the truth of \( A \) permits the truth of \( B \)’ is false just when the truth of \( A \) is sufficient for the falsity of \( B \).

The following contraposed forms of the basic positive principles for implication are derivable using the negation principles so far adduced:

\[
NS & \quad \neg(A \rightarrow B \& C) \Rightarrow \neg(A \rightarrow B) \lor \neg(A \rightarrow C)
\]

\[
NS \lor \quad \neg(A \lor B \rightarrow C) \Rightarrow \neg(A \rightarrow C) \lor \neg(B \rightarrow C)
\]

\[
NS \circ \quad \neg(A \rightarrow B \circ C) \Rightarrow \neg(A \rightarrow B) \land \neg(A \rightarrow C)
\]

\[
NS \oplus \quad \neg(A \oplus B \rightarrow C) \Rightarrow \neg(A \rightarrow C) \lor \neg(B \rightarrow C)
\]
CHAPTER 3. NEGATION

\[ NS\forall \neg(A\to(\forall x)B)\Rightarrow(\exists x)\neg(A\to\neg B) \] (where \( x \) is not free in \( A \)).

\[ NS\exists \neg((\exists x)B\to A)\Rightarrow(\exists x)\neg(B\to\neg A) \] (where \( x \) is not free in \( A \)).

\[ NM\to_1 \neg B\Rightarrow\neg C \vdash \neg(A\to B)\Rightarrow\neg(A\to C) \]

\[ NM\to_2 A\Rightarrow B \vdash \neg(A\to C)\Rightarrow\neg(B\to C) \]

Similarly, we can derive the contraposed forms of all the basic positive principles for permission.

Note also that \( A\otimes B\Rightarrow\neg(A\to\neg B) \) (where \( \otimes \) is either \& or \( \lor \)) will be derivable in systems with the \( F \) principle.

3.4 Hierarchy of Systems with Negation

The language of the negation systems in our main hierarchy will have all of the connectives of \( L \) (including \&, + and \( \neg \); quantified systems will of course have \( \forall \) and \( \exists \) as well). The basic systems, \( \mathcal{E} \) and \( \mathcal{QE} \), will add the negation principles \( N\& \), \( N\lor \), \( N\neg \), \( N\to \) and \( N\sim \) (plus \( N\forall \) and \( N\exists \) for \( \mathcal{QE} \)) to the positive bases, \( \mathcal{E}_+^{\neg\neg} \) and \( \mathcal{QE}^+_\neg\neg \). The only extensions to these basic systems we will be considering here are those with the \( D \), \( T \), \( E \) and \( F \) principles, which will be indicated by superscripts in the normal way.

3.5 Some Properties of Systems with Negation

All results given here for propositional systems \( \mathcal{E}^\alpha \) are to be understood as holding equally for their quantified counterparts \( \mathcal{QE}^\alpha \).

3.5.1 Negation Normal Form

The first result is that negation is essentially eliminable from \( \mathcal{E}^\alpha \), in the sense that we can define a normal forming operation \((.)^D \) on formulas such that
• \( A \leftrightarrow AD \) is provable in \( \mathcal{E}^\circ \) and

• \( A \Rightarrow B \) is provable in \( \mathcal{E}^\circ \) iff \( AD \Rightarrow BD \) is provable in \( \mathcal{E}^\circ_+\sim \).

The operation is a generalization of the standard 'DeMorganization' procedure in which negations are pushed in as far as possible so that they only occur in the context \( \neg p \), where \( p \) is an atomic sentence. At the \( \mathcal{E}^\circ_+\sim \) level, we make the trivial extension to the language to allow \( \neg p \) as an atomic sentence \( p \); these forms act in all ways just like new atoms and the negation sign has no logical effect in the positive systems. In other words, we could consistently replace all references to \( \neg p \) by some new atomic sentence \( q \) without changing any logical properties.

**Definition 5** A wff is in Negation Normal Form (NNF) iff \( \neg \) only occurs in the context \( \neg p \) where \( p \) is an atomic sentence. Additionally, we will often call a wff in NNF an NNF.

The translation is as follows:

• \( p^D = p \) for all atoms \( p \).

• \( (A \oplus B)^D = AD \oplus BD \) where \( \oplus \) is any binary connective.

• \( (QA)^D = Q(AD) \), where \( Q \) is any quantifier.

• \( (\neg\neg A)^D = AD \)

• \( \neg(A \oplus B) = (\neg A)^D \otimes (\neg B)^D \) where \( \oplus \) is \&, \lor, \circ or + and \( \otimes \) is respectively \lor, \&, + or \circ.

• \( (\neg(\forall x)A)^D = (\exists x)(\neg A)^D \)

• \( (\neg(\exists x)A)^D = (\forall x)(\neg A)^D \)

• \( (\neg(A \rightarrow B))^D = AD \sim (\neg B)^D \)

• \( (\neg(A \leftarrow B))^D = AD \rightarrow (\neg B)^D \)
CHAPTER 3. NEGATION

Clearly the translation will transform any wff into a NNF.

Theorem 43 $A \iff A^D$ is provable in $\mathcal{E}^\alpha$

The proof is a straightforward induction on the complexity of $A$.

Theorem 44 (Negation/NNF Reduction) $A \rightarrow B$ is provable in $\mathcal{E}^\alpha$ iff $A^D \rightarrow B^D$ is provable in $\mathcal{E}^\alpha_{\rightarrow}^+$. 

Proof Right to left follows from the previous theorem, noting that anything provable in $\mathcal{E}^\alpha_{\rightarrow}^+$ will also be provable in $\mathcal{E}^\alpha$. To show left to right, we demonstrate a procedure for converting a proof of $A \rightarrow B$ in $\mathcal{E}^\alpha$ into a proof of $A^D \rightarrow B^D$ in $\mathcal{E}^\alpha_{\rightarrow}^+$.

- Replace any axiom $A \rightarrow B$ by $A^D \rightarrow B^D$. It is easy to see that for all the positive axioms $A \rightarrow B$, $A^D \rightarrow B^D$ is just another instance of the same axiom form. For the negative axioms, $A^D = B^D$ and of course $A^D \rightarrow A^D$ is provable in $\mathcal{E}^\alpha_{\rightarrow}^+$.

- Assume that $A \rightarrow B$ follows from one or more other entailments by an application of some $\mathcal{E}^\alpha$ derivation rule, and that for each such antecedent $C \rightarrow D$, $C^D \rightarrow D^D$ is provable in $\mathcal{E}^\alpha_{\rightarrow}^+$. It is easy to see that the application of the same rule in $\mathcal{E}^\alpha_{\rightarrow}^+$ using the NNF antecedents will give a $\mathcal{E}^\alpha_{\rightarrow}^+$ proof of $A^D \rightarrow B^D$.

3.5.2 Conservative Extensions

As consequences of Negation/NNF reduction we have the following conservative extension results:

Theorem 45 $\mathcal{E}^\alpha$ is a conservative extension of $\mathcal{E}^\alpha_{\rightarrow}^+$

Proof Follows straightforwardly from Negation/NNF reduction, since for any $A$ in the language of $\mathcal{E}^\alpha_{\rightarrow}^+$, $A^D$ is clearly the same as $A$. 
CHAPTER 3. NEGATION

Theorem 46 $E^\alpha$ is a conservative extension of $E^\alpha\beta$, where $\beta$ is either null or the superscript of any linguistic extension to the basic vocabulary: that is, $\circ$, $\circ\rightarrow$, $+$, $\rightarrow$ or $\rightarrow$.

Proof Follows immediately from the conservative extension results for the positive systems and the previous theorem.

3.5.3 Contraposition

Theorem 47 (Admissability of Contraposition for the Basic System) If $A\rightarrow B$ is provable in $E$ then so is $\neg B\rightarrow \neg A$.

Proof By the normal induction on the construction of the entailment system. Details left to the reader.

We will now demonstrate that this result does not extend to extensions to $E$ with $D$, $T$, $E$ or $F$; see Sec. 3.7 for a discussion of and attempted justifications for this curious situation. Consider the consequences if the following are not provable in $E^\alpha$ (the restriction of $E^\alpha$ to the $\&$ and $\lor$ vocabulary of $L$, with $\neg$ introduced in an inactive way as discussed above), where $p$, $q$ and $r$ are distinct atomic sentences:

\begin{enumerate}
  \item $\neg q \rightarrow \neg p \lor p\&\neg q$
  \item $p\&\neg r \rightarrow p\&\neg q \lor q\&\neg r$
  \item $q\&\neg r \rightarrow \neg p \lor p\&q\&\neg r$
\end{enumerate}

By Th. 17, none of these would be provable in $E^\alpha$ for any $\alpha$, and similarly for the corresponding forms with intensional conjunction and disjunction instead of their extensional counterparts. By Ths. 6 and 7, none of the following will be provable in systems with permission:

- $\neg q \rightarrow \neg p \lor (p \rightarrow \neg q)$
- $p \rightarrow \neg r \rightarrow (p \rightarrow \neg q) \lor (q \rightarrow \neg r)$
• \( q \rightarrow r \Rightarrow \neg p \vee (p \neg q \rightarrow \neg r) \)

and so finally, by Th. 46 (and using \( N \rightarrow \) and other negation principles), none of the following will be provable in any of our negation systems \( E^{\alpha} \):

• \( \neg q \Rightarrow \neg (p \neg (p \rightarrow q)) \)

• \( \neg (p \rightarrow r) \Rightarrow \neg ((p \rightarrow q) \neg (q \rightarrow r)) \)

• \( \neg (p \rightarrow r) \Rightarrow \neg (p \neg (p \neg q \rightarrow r)) \)

These are of course just instances of the contraposed forms of D, T and E, and so the conclusion we reach is that if a, b and c are not provable in \( E^{+z} \), contraposition is not admissible for any entailment system \( E^{\alpha} \) where \( \alpha \) includes any of these elimination principles.

The following results are easily demonstrated:

• a is provable in \( E^{+z} \) iff \( \neg q \Rightarrow \neg p \vee p \) and \( \neg q \Rightarrow \neg p \vee \neg q \) both are.

• b is provable iff \( p \& \neg r \Rightarrow q \vee \neg q, p \& \neg r \Rightarrow p \vee q, p \& \neg r \Rightarrow p \vee \neg r \) and \( p \& \neg r \Rightarrow \neg q \vee \neg r \) are.

• c is provable iff \( q \& \neg r \Rightarrow \neg p \vee p, q \& \neg r \Rightarrow \neg p \vee q \) and \( q \& \neg r \Rightarrow \neg p \vee \neg r \) are.

The provability of each of a, b and c therefore rests on the provability of an entailment of the form \( A \Rightarrow p \vee \neg p \) where neither \( p \) nor \( \neg p \) occur in \( A \); and we know that such entailments are not provable, because of the weak relevance result of Th. 1.

Thus we have shown:

**Theorem 48** Where D, T or E is in \( \alpha \), Contraposition is not admissible for \( E^{\alpha} \).

An instance of the contraposed form of the \( F \) principle is

\( \neg (p \rightarrow \neg p) \Rightarrow \neg (p \& \neg p) \)
where $p$ is an atomic sentence and $\otimes$ is either $\circ$ or $\&$. Equivalently,

$$p \rightarrow p \rightarrow p \oplus p$$

where $\oplus$ is either $+$ or $\vee$. By Th. 45, this is provable in $E_\alpha$ iff it is provable in $E_+^{+\rightarrow}$; if it were indeed provable, $\neg p \oplus p$ would be an element of $L_+^{+\rightarrow}$ since of course $p \rightarrow p$ is. But the metavaluations of Th. 25 and Th. 27 tell us on the one hand that if this were an element of the logic, then either $p$ or $\neg p$ would be also, and on the other that no atom is an element of the logic, and hence we have:

**Theorem 49** Where $F \in \alpha$, Contraposition is not admissible for $E_\alpha$.

### 3.5.4 Replacements

Using the admissibility of Contraposition for $E$, the result of Th. 3 extends easily to the basic negation system:

**Theorem 50 (Replacement of Equivalents)** If $A \iff B$ is provable in $E$, and $D$ is like $C$ except for substituting an occurrence of $B$ for some occurrence of $A$, then $C \iff D$ is provable also.

The most we are able to show for extended negation systems is the partial result of Th. 52 below.

**Theorem 51** Where there is an occurrence of a formula $A$ in another formula $B$, there is a corresponding occurrence of either $A^D$ or $(\neg A)^D$ in $B^D$.

**Proof** By induction on the complexity of the occurrence of $A$ in $B$, defined as follows:

- The occurrence of $A$ in itself has a complexity of 0.
- If the complexity of the occurrence of $A$ in $B$ is $n$, then the complexity of the corresponding occurrence in $B \bullet C$, $C \bullet B$ and $QB$ is $n + 1$, where $\bullet$ is
any binary connective and \( Q \) is any quantifier; and the complexity of the corresponding occurrence in \( \neg B \) is again \( n \).

The induction proceeds as follows:

**Basis** If the complexity of the occurrence of \( A \) in \( B \) is 0 then \( B \) is \( \neg^i A \), where \( \neg^i \) is a string of \( i \geq 0 \) negation signs. The only clause in the NNF translation which has any effect on the result here is \( (\neg^j A)^D = A^D \), and clearly \( B^D \) will be \( A^D \) in the case where \( i \) is even, and \( (\neg A)^D \) when it is odd.

**Inductive Step** Show that the statement holds for a complexity of \( n + 1 \) under the assumption that it holds for \( n \). Assume that the occurrence of \( A \) in \( B \) has complexity \( n \); then it has complexity \( n + 1 \) in \( \neg^i (B \cdot C) \), \( \neg^i (C \cdot B) \), and \( \neg^i Q B \). The result then follows by the inductive hypothesis and the various clauses of the NNF translation.

**Definition 6 (True and False Occurrence)** An occurrence of \( A \) in \( B \) is a true (false) occurrence iff the corresponding occurrence in \( B^D \) is \( A^D \) (\( (\neg A)^D \)).

The partial replacement result is then:

**Theorem 52 (True and False Replacement)** Let \( D \) be like \( C \) except for replacing an occurrence of \( A \) by an occurrence of \( B \). Then

- If \( A \) occurs truly in \( B \), and \( A \iff B \) is provable in \( E^a \), then so is \( C \Rightarrow D \).
- If \( A \) occurs falsely in \( B \), and \( \neg A \iff \neg B \) is provable in \( E^a \) then so is \( C \Leftarrow D \).

**Proof** If \( A \) occurs truly in \( C \), then \( A^D \) occurs in \( C^D \). If \( A \iff B \) is provable in \( E^a \), then by Th. 43 \( A^D \iff B^D \) is provable in \( E^a_+ \). If \( D \) results from \( C \) by replacing this occurrence of \( A \) by \( B \), then clearly \( D^D \) results from \( D \) by replacing the corresponding occurrence of \( A^D \) by \( B^D \). By Th. 3, \( C^D \iff D^D \) is provable in \( E^a_+ \) and so \( C \iff D \) is provable in \( E^a \), by Th. 44. A similar proof establishes the result for falsely-occurring \( A \).
It is easy enough to demonstrate that \( D, T, E \) and \( F \) level systems do not have full replacement properties. For example, 
\[ q \iff (p \& (p \rightarrow q)) \vee q \] 
is provable at the \( D \) level, but 
\[ \neg q \iff (\neg p \vee (p \rightarrow \neg q)) \& \neg q \] 
is not; if it were, \( \neg q \rightarrow \neg p \vee (p \rightarrow \neg q) \) would be also, contradicting the results of the last section.

### 3.6 Some Properties of Logics with Negation

We leave it to readers to convince themselves that all the results of Sec. 2.10.2 extend \textit{mutatis mutandis} to the negation systems.

**Theorem 53** (Positive Representability) \( A \in \mathcal{L}_\alpha \) if and only if \( A^D \in \mathcal{L}_+^{+\neg} \).

\textbf{Proof} By Th. 24, \( A \in \mathcal{L}_\alpha \) iff \( Id \vdash A \) is provable in \( \mathcal{S}_\alpha \) (where by \( Id \) we mean the conjunction of all instances of Identity in the full language \( L \)). By Th. 44, \( Id \vdash A \) is provable in this system iff \( Id^D \vdash A^D \) is provable in \( \mathcal{E}_+^{+\neg} \). It is straightforward to show that \( Id_+ \iff Id^D \) is provable in \( \mathcal{E}_+^{+\neg} \) (where by \( Id_+ \) we mean the conjunction of all instances of Identity in the positive part of \( L \), augmented with an 'inactive' \( \neg \) occurring only in atomic contexts). Hence \( Id_+ \vdash A^D \) is provable, and the result follows by Th. 24 once again.

**Theorem 54** (Conserving Positive Logics) \( \mathcal{L}_\alpha \) is a conservative extension of \( \mathcal{L}_+^{+\neg} \).

\textbf{Proof} Immediate from the previous theorem; for if \( A \) is a sentence of the (augmented) positive part of \( L \), it is clearly the case that \( A^D = A \).

Given the conservative extension results for the positive logics, it immediately follows that \( \mathcal{L}_\alpha \) is a conservative extension of \( \mathcal{L}_+^{\beta \neg} \) where \( \beta \) is either null, \( +, \neg, o \rightarrow \) or \( + \rightarrow \).

**Theorem 55** (Entailment Normality of Non-Enthymeme Logics) \( E \notin \alpha \), \( \mathcal{L}_\alpha \) is entailment normal.
Proof Let \( A \rightarrow B \in \mathcal{L}^\alpha \). Then by Th. 53, \( A^D \rightarrow B^D \in \mathcal{L}^+_{\rightarrow} \), and so \( A^D \Rightarrow B^D \) is provable in \( \mathcal{E}^+_{\rightarrow} \). The result follows by Th. 44.

Theorem 56 Entailment and Logical Implication in Enthymeme Logics Where \( E \in \alpha \), \( A \rightarrow B \in \mathcal{L}^\alpha \) iff \( A \circ \text{Id} \Rightarrow B \) or \( A \Rightarrow B \) is provable in \( \mathcal{E}^\alpha \). Where \( E, D \in \alpha \), \( A \rightarrow B \in \mathcal{L}^\alpha \) iff \( A \circ \text{Id} \Rightarrow B \) is provable in \( \mathcal{E}^\alpha \).

Proof Right to left is straightforward in each case. So let \( A \rightarrow B \in \mathcal{L}^\alpha \). Then by Th. 53, \( A^D \rightarrow B^D \in \mathcal{L}^+_{\rightarrow} \). By Th. 27, \( A^D \Rightarrow B^D \) or \( A^D \circ \text{Id} \Rightarrow B \) is provable in \( \mathcal{E}^+_{\rightarrow} \); clearly \( \text{Id}^D \iff \text{Id}^D \) is provable in this system also, and the first result follows by simple moves and Th. 43. The second result relies in the same way on Th. 29.

3.7 Discussion

The major objections to our negation principles (besides those which might be levelled against a paraconsistent negation as such) will centre around the inadmissibility of Contraposition for the strong systems, perhaps the most serious consequence being the lack of full replacement properties (see Sec. 3.5.4). Here we will attempt to address the issues, and show why we think the most plausible formulations in which Contraposition is admissible are intuitively inferior. As we noted in Sec. 3.1, if these arguments fail to convince we think that the development given in the preceding sections at least gives a coherent characterization of the theoretical basis of the logic programming system described in the next Chapter, even if this basis is thought to be incomplete. We believe, however, that our negation is quite well motivated as a very weak form of ‘not’, and possibly better-motivated than the standard relevant reading as a paraconsistent operator.

3.7.1 Negation in Standard Relevant Logics

The clearest deconstruction of the intuitions underlying standard relevant negation we know of is to be found in [SMPB82, §2.9], which lists the following as
fundamental properties for a good negation, without much further argument:

- Double Negation.
- DeMorgan type principles.
- Contraposition, at least in rule form: $A \Rightarrow B \vdash \neg B \Rightarrow \neg A$
- Residual exclusion/exhaustion principles, such as $A \lor \neg A$ (Excluded Middle) or $\neg(A \land \neg A)$ (which is equivalent to Excluded Middle given DeMorgan's Laws) for every regular theory.

Our negation principles only satisfy the first two requirements; we take De Morgan's Laws and Double Negation to be essential features, but not Rule Contraposition or Excluded Middle. It is not to be expected that an argument over the relative merits of the two proposals could ever be settled definitively ('this one is closer to the Truth than that one') because to a certain extent we must be dealing with different selections from the general pool of intuitions surrounding the word 'not'. The most we can hope for is that the principles we have developed for negation will be seen as plausible given our reading of truth and falsity as independent qualities.

The question we pose is this: What intuitions about negation, apart from assumptions of exclusive and exhaustive truth and falsity, will lead us to believe in Contraposition? Given exclusiveness and exhaustivity, Contraposition follows straightforwardly: the 'content' of $\neg A$ is just the complement of the content of $A$, and so clearly whenever $B$ is contained in $A$, $\neg A$ will be contained in $\neg B$. But having abandoned this view of negation, what fundamental properties of truth and falsity can we posit coherently, which will lead us to the same conclusion? The view of [SMPBS2] is clear; Contraposition is just a fundamental characteristic of anything which deserves to be called a negation, and the relationship between truth and falsity is defined by the properties listed above. There is nothing incoherent about this approach, but we are surely to be forgiven if we persist in asking the question: Why is Contraposition a fundamental property? Of course, the same question could be put with regard to DeMorgan's Laws and Double
CHAPTER 3. NEGATION

Negation, and we have no definitive answer. What we can do is show how a 'bottom-up' approach taking the basic notion of truth and falsity as independent will deliver Contraposition if we add some principles which while not required by this interpretation, are not unreasonable for everyday purposes. We have then the possibility of arguing that our negation is more basic than the standard relevant one, since it is rooted in a simple premiss about the nature of truth and falsity and provides a framework within which the more orthodox negation can be explained.

In the relevant logic world there are two semi-rival views on how best to explain negation; [MM86] calls these the 'American Plan' and the 'Australian Plan'. Our motivation in Sec. 3.1 puts us fair and square in the Yankee camp, and we prefer to view the development of this chapter as a way of extending, to the general case, the American ideas of [Dun76], [Bel77a], [Bel77b] and [BD81], which focus mainly on the first-degree implicational fragment. Although we want to avoid becoming embroiled in the dispute between the two, it will be instructive to give a brief comparison, and this is best accomplished at the first-degree implicational level (see [RR79] for the Australian version; [Bol85] gives an American account, in the same terms, which can be extracted straightforwardly from [Dun76] and [BD81]). Both Plans deliver the same logic (i.e., $R_{fde}$), but there are important differences in the way they motivate negation. Model structures in both cases include a set of worlds $K$ and a set of first-degree implications $R$. Elements $s \in K$ can be thought of as either interpretations, mapping sentences into truth-values, or just as sets of sentences, whichever is preferred. We will take the latter view here. Each $s$ is thought of in the first instance as an unrestricted set of atoms; there is no $p$ which must be an element of any $s$, and there is no restriction against having both $p$ and $\neg p$, or neither $p$ nor $\neg p$. The membership condition is extended to general sentences in the $\& \lor \neg$-vocabulary with recursive conditions; both Plans have the same conditions for $\&$ and $\lor$:

- $A \& B \in s$ iff $A \in s$ and $B \in s$
- $A \lor B \in s$ iff $A \in s$ or $B \in s$
The Plans differ in their treatment of negation. The American conditions for negation are like those we gave in Sec. 3.2:

- \( \neg A \in s \) iff \( A \in s \)
- \( \neg (A \& B) \in s \) iff \( \neg A \in s \) or \( \neg B \in s \)
- \( \neg (A \lor B) \in s \) iff \( \neg A \in s \) and \( \neg B \in s \)

(It is easy enough to see that for any \( A \), there is no bar to having both \( A \) and \( \neg A \) as elements of \( s \), and it is possible for neither \( A \) nor \( \neg A \) to be in \( s \).) Australian model structures, on the other hand, deal with negation by means of an operation \( '•' \) on elements of \( K \); for each \( s \in K \), \( s^* \in K \) as well, and \( s^{**} = s \) for every \( s \).

The condition for negation is then:

- \( \neg A \in s \) iff \( A \notin s^* \)

The American conditions all follow from this; for example, \( \neg (A \& B) \in s \) iff \( A \& B \notin s^* \) iff \( A \notin s^* \) or \( B \notin s^* \) iff \( \neg A \in s \) or \( \neg B \in s \). The requirements on \( R \) are also different; the Australian version is a straightforward S5-style strict-implication condition:

- \( A \rightarrow B \in R \) iff \( B \in s \) whenever \( A \in s \) for all \( s \in K \).

We will have \( \neg B \rightarrow \neg A \in R \) whenever \( A \rightarrow B \in R \): Let \( A \rightarrow B \in R \) - then \( B \in s \) whenever \( A \in s \) for all \( s \in K \), and so \( A \notin s \) whenever \( B \notin s \); but \( s = s^{**} \), and so \( \neg A \in s^* \) whenever \( \neg B \in s^* \), for all \( s \in K \), and the result follows by the closure of \( K \) under \( '•' \). The American Plan, in contrast, must make explicit provision for contraposed forms:

- \( A \rightarrow B \in R \) iff \( B \in s \) whenever \( A \in s \) and \( \neg A \in s \) whenever \( \neg B \in s \), for all \( s \in K \).

Both Plans have characteristics which appear odd (to us, at least): specifically, the \( '•' \) operation in the Australian Plan, and the double-barreled condition for
implication in the American. In [SMPB82, §2.9], the * operation is motivated partly as a way of it providing enough 'exclusiveness-nature' for negation to deliver Contraposition (as well as DeMorgan's Laws and Double Negation) without forcing total exclusion; it thus reflects in a formal way the historical motivation for negation in relevant logics - to preserve as much of classical 'not' as possible while avoiding paradox. Again this is quite coherent with respect to a development which takes the properties listed above as the fundamental definition of the relationship between truth and falsity.

We confess to being among those in whose hearts the * operation does not kindle a warm glow, partly because it does not fit very well with the elegant algebraic picture which Belnap paints in [Bel77a] and [Bel77b]. There we are invited to think of the database of our reasoner as corresponding to an American first-degree model. The set K becomes an epistemic state, and for each formula A in the & V ¬-vocabulary we can define operations $A^+$ on epistemic states for which $K' = A^+(K)$ is the minimal mutilation of K which makes A true in K'. By the locution 'A true in K'' we mean that $A \in s$ for all $s \in K'$. The empty database can be represented by the set of all s which obey the American conditions, and $A^+(K)$ is $K \cap \text{Tset}(A)$, where $\text{Tset}(A)$ is the set of all s s.t. $A \in s$. For first-degree implications $A \rightarrow B$ we can also define operations $(A \rightarrow B)^+(K) = K'$ where $K'$ is the minimal monotonic mutilation of K for which $A \rightarrow B \in R$ for the corresponding American model $(K', R)$. How does this work on the Australian Plan? First note that we cannot use an Australian K' to represent an inconsistent database directly, for the * operator will force us to have $\lnot A \not\in s$ for all $s \in K$ whenever $A \in s$ for all $s \in K$. The only solution which springs to mind is one which tends to undermine the Australian nature of the model: instead of K, take two sets Q and $Q^*$ s.t. for all $s \in Q$ there is an $s^* \in Q^*$, and vice-versa, with the condition that $s^{**} = s$. When we assert A to the system, Q is minimally mutilated to $A \in s$ for all $s$ in the new Q, and $Q^*$ is changed to whatever extent is required by the * operator. Note that we do not have $A \in s$ for all $s$ in the new $Q^*$; instead we have $\lnot A \not\in s$ for all such s. Q represents our database, and $Q^*$ represents whatever is not denied by it: $A \in s$ for all $s \in Q$ iff $\lnot A \not\in s$ for all $s \in Q^*$.
CHAPTER 3. NEGATION

for all $s \in Q^*$. So far there has been no essential change to the Australian Plan; we have just designated a $Q \subseteq K$ to represent the actual database, so that changes to the database are effected in the first instance via changes to $Q$. However, if we now ask what the effect of an implicational assertion $A \rightarrow B$ is to be, the situation is more complicated. For if the answer is just to change $Q$ to $Q'$ in such a way that $B \in s$ whenever $A \in s$, the result is not an Australian model with $A \rightarrow B \in R$, because changing $Q$ in this way need not produce the same change in $Q^*$; if $B \in s$ whenever $A \in s$ for all $s \in Q$ then $\neg A \in s$ whenever $\neg B \in s$ for all $s \in Q^*$, but that is as far as it goes. To get an Australian model we could stipulate that adding $A \rightarrow B$ to the system changes $Q^*$ in the same way as $Q$: $B \in s$ whenever $A \in s$ for all $s \in Q^*$. But this seems unjustified, when we recall that for a non-implicational assertion $A$, only $Q$ changed to make $A$ true, whereas $Q^*$ changed to make $\neg A \notin s$ for all $s \in Q^*$. Alternatively, we could stipulate that whenever we add $A \rightarrow B$, not only do we require $B \in s$ whenever $A \in s$ for all $s \in Q$, but also $\neg A \in s$ whenever $\neg B \in s$, but this just gives us what seems to be an unnecessary complication of the American Plan; specifically, it is as if we had taken an American model and added a new set of worlds $K^*$, induced from $K$ by the $\ast$ operator.

This argument is intended only to illustrate our personal reasons for not liking the Australian Plan; see [Cop79] for other attacks on it of varying worth, and [MM86] for a defence. The major point we wish to make is that in both models, the desire to retain contraposition for implication necessitates the inclusion of elements which are at least odd-seeming.

So what happens if we remove the odd element from the American Plan, replacing its condition on $R$ with the simpler Australian condition? In terms of the logic, the answer, perhaps surprisingly, is nothing. A first degree implication $A \rightarrow B$ is valid (i.e., logically true) under the American Plan just when it is an element of $R$ for every American model $(K, R)$. It is easy to see that this amounts to saying that $A \rightarrow B$ is valid when for every $s$ which obeys the American conditions, $B \in s$ whenever $A \in s$ and $\neg A \in s$ whenever $\neg B \in s$. What will not be immediately apparent is the equivalence of this condition with each of its two ‘halves’, since
we can show the following:

\[ B \in s \text{ whenever } A \in s \text{ for every } s \text{ obeying the American conditions, } \]

if and only if \( \neg A \in s \) whenever \( \neg B \in s \) for every such \( s \).

See [Dun76] or [Bol85] for proofs of this assertion. So if we replace the American double-barreled condition on \( R \) with the Australian version, we will certainly get a different class of models (specifically, ones which do not have \( \neg B \rightarrow \neg A \in R \) whenever \( A \rightarrow B \in R \)), but the set of valid implications will remain the same. Note the correspondence with our development in this Chapter, where Contraposition is not taken as a fundamental principle but is admissible, as a consequence of our other conditions for negation, in the basic system \( \mathcal{E} \). In [Bol85] we argued that these modified American models were at least as well-motivated as the original forms, and gave arguments against Contraposition which were less-well developed versions of those we shall present here. We are encouraged by the implicit agreement of Blair and Subrahmanian in [BS87]. The fact that our logic at the first-degree level (which is the only level for which most of us would claim to have very firmly rooted intuitions) does not alter when we discard Contraposition as an explicit principle should make more plausible the consideration of full logics for which it is not admissible.

### 3.7.2 Contraposition Via Suppression

Here we will illustrate what we believe to be the intuitions underlying Contraposition, and show why we do not believe they are not essential for a non-exclusive and non-exhaustive negation. Consider the following \( \mathcal{E} \) proof of \( \neg B \circ (A \rightarrow A) \Rightarrow \neg A \circ (B \circ \neg B) \) from \( A \Rightarrow B \):

\[
\neg B \circ (A \rightarrow A) \Rightarrow \neg B \circ (B \rightarrow A) \quad \text{(using } A \Rightarrow B \text{ and Monotonicity principles)}
\]

\[
\Rightarrow \neg A \circ B \circ \neg B \quad \text{(using } D \circ \text{ and } C \circ \text{).}
\]

Were we to have the following Suppression principles

\[
S f \quad A \circ (B \circ \neg B) \Rightarrow A
\]
Contraposition would follow immediately, and in fact adding these to $\mathcal{E}_+^{DTE}$ will give us the full relevant logic $R$, as we show in Sec. 3.8. Note that the corresponding extensional forms

- $A \lor (B \& \neg B) \Rightarrow A$
- $A \Rightarrow A \& (B \lor \neg B)$

would allow the two Lewis arguments to go through, thus introducing irrelevance, whereas $S_f$ and $St$ do not (since we do not have Weakening for the intensional connectives). In some sense, therefore, $St$ and $S_f$ can be seen as bringing to the system as much of classical negation as will not produce irrelevance. $S_f$ says that we can suppress a contradiction from a disjunction when the disjunction is between real alternatives; $St$ that we can introduce the effects of an Excluded Middle in arbitrary contexts without being allowed to deduce the Excluded Middle itself. Our intuition is to be more suspicious of $St$ than $S_f$, just because $St$ looks like an elliptical form of $A \circ (B \rightarrow \neg B) \Rightarrow A \circ (B \rightarrow \neg B)$. As we will show in Sec. 3.8, adding just $S_f$ to $\mathcal{E}_+^{DTE}$ will produce a system for which the corresponding complete logic (i.e., containing all instances of $A \rightarrow A$) is again equivalent to $R$; this is our preferred way of viewing $R$. Note that in systems with Detachment, Excluded Middle will follow from $St$: $A \rightarrow A \Rightarrow (A \rightarrow A) \circ (A \rightarrow A)$ (by $St$) $\Rightarrow A \circ (A \rightarrow A) \rightarrow A$ (by $Do+$) $\Rightarrow A \rightarrow A$ (by $Do$). However, this seems to us to be something of a trick and we prefer to view Excluded Middle as we do in the complete logics, as a broadening of the notion of a necessary truth from the implicational basis.

So let us turn to $S_f$. Our objection to it, as a fundamental principle, is just the same objection that we (and presumably most explicators of relevant paraconsistent logics) have to its extensional counterpart; given that it is at least possible for $B$ and $\neg B$ both to hold, what grounds do we have for concluding $A$ from a real choice between $A$ and $B \& \neg B$ (a move of course allowed by $S_f$)? As a principle of everyday reasoning, $S_f$ is of course quite natural, just because in normal situations we are safe in tacitly assuming that both $B$ and $\neg B$ do not hold. But
CHAPTER 3. NEGATION

there seems to us no good reason for taking it as a fundamental principle, if we assume the possibility of inconsistency.

3.7.3 Rejecting Contraposed Forms of Extended Principles

Evidently, Contraposition would be admissible for all systems if we were to take the following as always paired with $D$, $T$, $E$ and $F$ respectively:

$ND \ A\Rightarrow B + (\neg B\Rightarrow A)$

$NT \ A\Rightarrow C \Rightarrow (A\Rightarrow B) + (\neg B\Rightarrow C)$

$NE \ B\Rightarrow C \Rightarrow A + (\neg A\circ B\Rightarrow C)$

$NF \ A\Rightarrow B \Rightarrow \neg A\circ B$

Readers may verify for themselves that these will deliver the contraposed forms of $D$, $T$, $E$ and $F$, given our other negation principles. The question we ask here is whether they are well motivated in and of themselves; we think the answer is ‘no’. $ND$, $NT$ are $NE$ can be characterized as introduction principles, because something appears in an intensional context on the right which was absent on the left, and this is the root of our objections to them; in $ND$, for example, why should the meaning of $A$ include any information about what some arbitrary $\neg B$ permits?

In fact $ND$ to our mind involves a clear instance of a suppressed usage of Excluded Middle: for $A\circ (B + \neg B) \Rightarrow B + \neg B\circ A$ (by $Do+$) and so $A\circ (B + \neg B) \Rightarrow B + (\neg B\Rightarrow A)$ in a system with the $F$ principle. Appealing more closely to intuitions, what reason could we have for concluding that ‘$B$ is true or the falsity of $B$ permits the truth of $A$’ from the truth of $A$, unless somehow or other we were assuming in the background that $B$ is always either true or false? And this of course conflicts with the non-elliptic nature of our entailments.

The situation for $NT$ and $NE$ is less clear, but our objections follow the same general lines. The derivation of $NT$ using Suppression principles is as follows:
CHAPTER 3. NEGATION

\[ A \sim C \Rightarrow (A \sim C) \circ ((A \rightarrow \sim B) + (A \sim B)) \circ ((\sim B \rightarrow \sim C) + (\sim B \sim C)) \quad (St) \]
\[ \Rightarrow (A \sim C) \circ ((A \rightarrow \sim C) + (A \sim B) + (\sim B \sim C)) \quad (To \text{ and } Do+) \]
\[ \Rightarrow (A \sim C) \circ (A \rightarrow \sim C) + (A \sim B) + (\sim B \sim C) \quad (Do+) \]
\[ \Rightarrow (A \rightarrow B) + (\sim B \sim C)(Sf) \]

We would probably be willing to agree that either \( A \) permits \( B \) or the lack of \( B \) permits \( C \) whenever \( A \) permits \( C \); for if \( A \) rules out \( B \) and the lack of \( B \) rules out \( C \), then presumably \( A \) rules out \( C \) also. But of course \( \sim B \) does not correspond to the ‘lack of \( B \)’, but rather to the presence of the falsehood of \( B \). The Suppression derivation for \( NE \) is as follows:

\[ B \sim C \Rightarrow (B \sim C) \circ (A \rightarrow \sim A) \circ ((\sim A \circ B \rightarrow \sim C) + (\sim A \circ B \sim C)) \quad (St) \]
\[ \Rightarrow (B \sim C) \circ ((B \rightarrow \sim C) + A \circ (\sim A \circ B \sim C)) \quad (E \text{ and } Do+) \]
\[ \Rightarrow (B \sim C) \circ (B \rightarrow \sim C) + A \circ (\sim A \circ B \sim C) \quad (Do+) \]
\[ \Rightarrow A \circ (\sim A \circ B \sim C)(Sf) \]

Given the lack of \( A \), and a requirement that the lack of \( A \) together with \( B \) rules out \( C \), then we would probably conclude (enthymematically) that \( B \) rules out \( C \). So if \( B \) permits \( C \), either \( A \) is present or the lack of \( A \) together with \( B \) permits \( C \). Again the argument will not go through when we substitute ‘presence of the falsity of …’ for ‘the lack of …’.

For \( NF \), we can accept that either \( A \) is not true or \( B \) is true, whenever \( A \rightarrow B \) is true, but there would seem to be no reason to conclude that either \( A \) is false or \( B \) is true in this situation. The Suppression argument is straightforward:

\[ (A \rightarrow B) \circ (\sim A + A) \Rightarrow \sim A + A \circ (A \rightarrow B) \Rightarrow \sim A + B. \] (Note that this argument shows that \( A \rightarrow B, \sim A + B \) will hold in complete logics with Detachment and Enthymeme).

3.7.4 Contraposition Via Alternative Falsity Conditions for Implication

As we mentioned above, in standard developments of relevance Contraposition principles are taken as fundamental, with DeMorgan’s Laws, for example, delivered as consequences. Our view is of course the opposite; DeMorgan’s Laws
(and Double Negation) are fundamental and for the basic system, Contraposition is admissible in virtue of these. The possibility remains that there is a better way of interpreting the falsity of $A \rightarrow B$ which will give us negation principles for implication that extend the admissibility of contraposition to the $D$, $T$ and $E$ levels.

Consider, then, the interpretation of $\neg(A \rightarrow B)$ which would be needed to independently justify this contraposed form of $D$:

$$\neg B \Rightarrow \neg A \lor \neg(A \rightarrow B).$$

The only reading which strikes us as being at all likely (and which still allows both $A \rightarrow B$ and $\neg(A \rightarrow B)$ to hold) is: ‘The falsity of $A$ is unnecessary for the falsity of $B$’ or in other words ‘$A$ not false permits $B$ false’.

$A$ must be either false or not false on our interpretation (which is not to say that it must be either false or true), so if $B$ is false, either $A$ is false or $A$ not false does not rule out $B$ false (and so it permits it): if $A$ were not false, and the lack of the falsity of $A$ did rule out the falsity of $B$, then clearly $B$ could not be false. (Notice that Contraposition is acceptable for our metatheoretic exclusive ‘not’.) The contraposed forms of the $T$ and $E$ are validated by this reading in similar ways.

On the other hand, the intuitively desirable consequence of our $F$ principle

$$A \otimes \neg B \Rightarrow \neg(A \rightarrow B)$$

(where $\otimes$ is either $\&$ or $\circ$) would no longer be validated, since there seems no good reason to conclude from $A$ and $\neg B$ that the lack of falsity of $A$ permits the falsity of $B$ (given non-exclusive and non-exhaustive truth and falsity). This seems to us a high price to pay. Note however that on the new proposal we would expect $\neg(A \rightarrow B)$ to follow from the lack of falsity of $A$ together with $\neg B$, on the same intuitions that gave us the $F$ principle. This would seem to be the correctly Contraposed form of what we can conclude from the truth of $A \rightarrow B$ (if we believed in Contraposition), namely that $A$ is not true or that $B$ is true.
CHAPTER 3. NEGATION

Formally, the new proposal has much to recommend it; it would bring to our systems the kind of symmetry which they so patently lack, giving us the expectation of being able to formulate tidy Gentzen systems and reasonably clean semantics of the normal kind for them. Our only problem is that we see no reason to believe in it, apart from these a prioristic, formal considerations: why should the fact that the falsity of $B$ is permitted by the lack of falsity of $A$ allow us to conclude that ‘The truth of $A$ is sufficient for the truth of $B$’ is false? If we must have Contraposition, then our opinion is that the correct explanation lies in the Suppression techniques of Sec. 3.7.2 which while not required by a paraconsistent reading of negation, do the job of introducing as much of the exclusiveness and exhaustiveness of everyday negation as possible while retaining relevance.

3.7.5 Possible Failure of Replacement Principles

As we pointed out above (following Th. 52), we do not have the principle of substitutivity of equivalents for our stronger systems. This principle is generally taken as an essential feature of a logic, and for many its failure will constitute the strongest argument against our systems, on grounds of incoherence. The provability of $A \leftrightarrow B$ in an entailment system amounts to a statement that $A$ and $B$ have the same logical meaning, and should therefore be freely intersubstitutable without alteration of logical properties; so the argument goes. However, with truth and falsity taken as essentially independent, we think that the partial results of Th. 52 are perhaps sufficient to establish the coherence of our systems in this respect. For the provability of $A \iff B$ in our systems is not to be interpreted as an assertion that $A$ and $B$ have the same meaning but rather that the truth of $A$ and the truth of $B$ do. Dually, the provability of $\neg A \iff \neg B$ amounts to a statement that the falsity of $A$ and the falsity of $B$ have the same meaning. In other words, for $A$ and $B$ to have completely the same meaning, both $A \leftrightarrow B$ and $\neg A \leftrightarrow \neg B$ would have to be provable. Given this, we should not expect to preserve equivalence when we substitute $B$ for an essentially false occurrence of $A$; as Th. 52 shows, equivalence will be preserved in this case if $\neg A \iff \neg B$ is provable, that is, when the falsity of $A$ and $B$ can be systematically identified.
CHAPTER 3. NEGATION

3.8 Suppression Systems

For the purposes of comparing our development with standard relevant logics, we will show how adding Suppression principles for negation to our stronger positive systems will deliver the full logic $R$. Because we do not take Contraposition to be a fundamental principle, the fit between the two developments is not very precise at weaker levels; although there are few other connections which could be drawn, we will content ourselves with the following:

Theorem 57 $\mathcal{L}$ is a conservative extension to the full ‘basic’ relevant logic $B$.

$B$ results from the Hilbert-style axiomatization of $B_\dagger$ of Sec. 2.10.7 by adding the axiom:

$$\neg\neg A \rightarrow A$$

and the rule

$$A \rightarrow \neg B \vdash \neg B \rightarrow A$$

(see [SMPB82, Sec. 4.1]). Details of proof left to the reader.

The Suppression systems we will consider are extensions to $\mathcal{E}^{+\text{DTE}}$ as follows:

- $\mathcal{E}^{S_1}$, which adds both $S_t$ and $S_f$.
- $\mathcal{E}^{S_2}$, which adds just $S_f$.

plus their quantificational counterparts (all results given here for the propositional systems are to be taken as applying equally to their quantified versions). These extensions also involve an extension to the language of $\mathcal{E}^{+\text{DTE}}$ with $\neg$. We will show that $R$ is equivalent to both $\mathcal{L}^{S_1}$ and $\mathcal{L}^{S_2}$ (recall that this is the smallest regular $\mathcal{E}^{S_1}$-theory which also contains every instance of $A \rightarrow \neg A$). Clearly these logics will share all of the properties of Enthymeme logics given in Sec. 2.10.5.

Theorem 58 The following are derivable in $\mathcal{E}^{S_1}$:
CHAPTER 3. NEGATION

- \( A \Rightarrow B \vdash \neg B \Rightarrow \neg A \)

- \( \neg (A \otimes B) \iff \neg A \oplus \neg B \) where \( \otimes \) is \&, \( \odot \), \( \lor \) or \( \div + \), and \( \oplus \) is respectively \( \lor \), \( + \), \& or \( \odot \).

- \( A \iff \neg \neg A \)

We proved the admissibility of Contraposition for entailments in the last Section; the proofs of the other two properties are left to the reader. We also have the following:

**Theorem 59** \( A \Rightarrow B \Rightarrow \neg A + B \) is provable in \( E^{S_1} \).

*Proof* \( A \Rightarrow B \Rightarrow (A \Rightarrow B) \odot (A \Rightarrow \neg A) \) (by \( St \)) \( \Rightarrow A \odot (A \Rightarrow B) \odot \neg A \) (by \( Do' \)) \( \Rightarrow \neg A + B \) (by \( Do, \) Monotonicity and Commutativity).

**Theorem 60** \( \neg A + B \Rightarrow A \Rightarrow B \in L^{S_2} \).

*Proof* \( A \odot (\neg A + B) \Rightarrow A \odot \neg A + B \Rightarrow B \) is provable in \( E^{S_2} \) using Distribution and \( Sf \).

The result follows by Exportation on the implicational counterpart of this.

Clearly \( L^{S_2} \subseteq L^{S_1} \), and taking these two results together, readers may satisfy themselves that the following are elements of \( L^{S_1} \) (where we use \( A \Rightarrow B \) as shorthand for \( (A \Rightarrow B) \& (B \Rightarrow A) \)):

- \( (A \Rightarrow B) \iff \neg A + B \)

- \( (A \Rightarrow B) \iff (\neg B \Rightarrow \neg A) \)

- \( \neg (A \Rightarrow B) \iff (A \odot \neg B) \iff (A \Rightarrow \neg B) \)

The full logic \( R \) results from the formulation of \( R_+ \) in Sec. 2.10.7 by adding the following axioms (see [AB75, §27.1]):

- **A12** \( A \Rightarrow \neg \neg A \)

- **A13** \( A \Rightarrow \neg B \Rightarrow B \Rightarrow \neg A \)
CHAPTER 3. NEGATION

If we define $A+B = \neg(\neg A \circ \neg B)$ in $R$ we have the following:

Theorem 61 $\mathcal{L}^{S_1} = R$

Proof To show that $R$ is contained in $\mathcal{L}^{S_1}$, the only addition we need to Th. 40 is to note that $A_{12}$ and $A_{13}$ are both contained in the latter, in virtue of Ths. 58, 59 and 60, and some simple moves. For the other direction, we need only show that $R$ is a regular $\mathcal{E}^{S_1}$-theory, and for this it is enough to show that it reifies $\mathcal{E}^{S_1}$ entailment. We leave the details to the reader.

Theorem 62 $(A \rightarrow B) \leftrightarrow \neg A + B \in \mathcal{L}^{S_2}$

Proof $\neg A + B \rightarrow A \rightarrow B \in \mathcal{L}^{S_2}$, by Th. 60. $(A \rightarrow) \circ (A + \neg A) \Rightarrow A \circ (A \rightarrow B) + \neg A$ (by $D_1$) $\Rightarrow B + \neg A$ (by $D_1$) $\Rightarrow \neg A + B$ (by $C_1$) is provable in $\mathcal{E}^{S_1}$. Exporting on the implicational form of this, we have $A + \neg A \rightarrow A + B \rightarrow \neg A + B \in \mathcal{L}^{S_2}$, and the result is immediate (since of course $A + \neg A \in \mathcal{L}^{S_2}$).

Theorem 63 $A \rightarrow \neg \neg A \in \mathcal{L}^{S_2}$

Proof In $\mathcal{E}^{S_2}$ we have $A \circ (\neg A + \neg A) \Rightarrow A \circ \neg A$ (by $D_1$) $\Rightarrow \neg A$ (by $S_1$). $\neg A + \neg A \rightarrow A + \neg A \in \mathcal{L}^{S_2}$ follows by Exporting on the implicational counterpart, and the result is immediate.

Theorem 64 $A \rightarrow \neg B \rightarrow B \rightarrow \neg A \in \mathcal{L}^{S_2}$

Proof Follows from Ths. 62 and 63.

Theorem 65 $\mathcal{L}^{S_2} = R$

Proof Clearly $\mathcal{L}^{S_2} \subseteq \mathcal{L}^{S_1}$. We also have $\mathcal{L}^{S_1} = \mathcal{L}^{S_1}$; by Th. 59, $A \rightarrow A \rightarrow \neg A + A \in \mathcal{L}^{S_1}$ so clearly it contains all instances of $A + \neg A$. So $\mathcal{L}^{S_2} \subseteq \mathcal{L}^{S_1} = R$ (by Th. 61). For the other direction, we need only show that $A_{12}$ and $A_{13}$ are elements of $\mathcal{L}^{S_2}$ and this follows by Ths. 63 and 64.
CHAPTER 3. NEGATION

Note that in $R$ we have $A \rightarrow B \leftrightarrow \neg A + B \leftrightarrow \neg (A \land B)$ so we can define intensional conjunction and disjunction in terms of implication. Note also that permission and fusion will coincide in this strong system, if we define $A \dashv \vdash B = \neg (A \rightarrow \neg B)$.

We can axiomatize $RQ$ by adding Q1, Q3 and Q6 from Sec. 2.10.7 to the axioms for $R$, and defining $(\exists x)A = \neg (\forall x)\neg A$. We leave it to readers to demonstrate the following:

Theorem 66 

- $RQ = \mathcal{Q}L^{S_1}$
- $RQ = \mathcal{Q}L^{S_2}$
Chapter 4

Relevant Logic Programming

4.1 Introduction

In this chapter we provide some computational models for and a description of the fragment of relevant logic computed by CLOGPROG ('Conditional Logic Programming'), our relevant logic programming system. We start with the computational details; they provide another vantage point from which to view relevant logics in general, and also serve to motivate the rather peculiar fragment actually computed. The major element in the translation between computation and logic is the treatment of conjunction; in the computation we use a conjunction sign ‘\&’ which will stand for extensional conjunction ‘\&’ in some positions and intensional fusion ‘\circ’ in others. The very natural computational structure which this ambiguous translation gives rise to should serve to lend it some support; later we show how it is more or less forced upon us if we want to retain a PROLOG-style evaluation scheme (and more generally, an extension to the SLD model which retains its essential characteristics). We also give some independent arguments for the plausibility of the translation.

Our logical treatment, couched mainly in terms of Gentzen formulations, deviates from the standard logic-programming preference for model theoretic and fix-point analysis. We have always been unclear about why the semantic consequence ‘\models’ is so often seen as a relation which ‘explains’ the provability relationship
CHAPTER 4. RELEVANT LOGIC PROGRAMMING

1, perhaps because our intuitions are not strongly algebraic\(^1\). The advantages of Gentzen formulations for our purposes are two-fold; they provide a logical description which is very close to the computational model, facilitating the proofs of the equivalence of the two, and they also provide a clear way of expressing the characteristics of our unusual fragment. We hope that this emphasis on proof theory will not detract too strongly from the interest of this Chapter for those whose intuitions lean the other way. We will be quite scrupulous about relating the Gentzen ‘\(\vdash\)' back to our semi-semantical ‘\(\Rightarrow\)' of the last two chapters, so this may serve as some consolation. On the other hand, we believe that familiarity with Gentzen systems is an essential part of the formal armament of anybody working in the area of computational logic.

Relevant logics in general are extremely expensive computationally. The current pinnacle of Automated Theorem Proving systems for relevant logics is the ‘Kripke’ system described in [TMM88], which is a complete theorem prover for the propositional part of the logic \(L_R\) (for which see Sec. 4.3). It incorporates many theoretical and pragmatic insights for reducing and pruning the search space, but is nevertheless too slow to form the basis for a logic programming system (besides being restricted to the propositional level). As an example, the current (Pascal) version of Kripke running on a Sun 3/50 takes 16 seconds (real time) to show that \(e\) follows from the set \(\{a, a \rightarrow b, b \rightarrow c, c \rightarrow d, d \rightarrow e\}\)\(^2\) whereas for a logic programming system the cost of this kind of operation has to be vanishingly small (in the order of milliseconds, at most). Any relevant logic programming language will have to sacrifice some expressive power for the sake of computational efficiency. Our goal has been to find a fragment which will minimize this loss while allowing a simple extension to the PROLOG computation scheme (and more generally, to SLD resolution). Our intention with CLOGPROG is to build a programming language, rather than an ATP system. By keeping our system close to the logic programming standard we expect that it will be able to share in much of the advanced work in the field (eg, techniques for parallel computation,\(^3\)

\(^{1}\)Bob Meyer has told us that proof theory, like a preference for white wine over claret, is a young man's folly; perhaps we are too young to understand this.

\(^{2}\)Data supplied in private communication by Paul Thistlewaite.
4.2 The Computation

We give three computational models in this section. The first, and most general, is an extension of the SLD model given in [Llo84]. As for SLD, the existence of a proof is independent of the particular computation rule chosen, and we capitalize on this fact to present simpler, but deductively equivalent structures which assume the standard PROLOG left-to-right scheme. The result is an elaboration of the computation tree model of [GR84], and gives a computational description which is easier to understand and structurally very close to the Gentzen systems used in the logical discussion of later sections. This will be our preferred model for discussing the computation; we include the SLD extension mainly for completeness, and to establish links with conventional logic programming developments. Finally, we have a ‘Conditional Query Evaluation Procedure’ (‘CQEP’) with deterministic candidate selection which is essentially a description of CLOGPROG’s algorithm (implemented by an extension to the simple stack-frame, structure-sharing interpreter described in [Hog84]).

4.2.1 Syntax

We will write ‘\( A \rightarrow B \)’ instead of the more normal (for logic programming) ‘\( B \leftarrow A \)’. We generalize the PROLOG notions of assertions and queries to allow for nested implications. Subformulas of a query may act like assertions in the course of a computation (and vice versa); for example, in a query \( A \rightarrow B \), \( A \) must have the form appropriate to assertions; dually, for an assertion \( A \rightarrow B \), \( A \) must have the form of a query.

The syntax is as follows:

1 An assertion has the form \( \forall A \) where \( A \) is an unquantified assertion and \( \forall \) is a string (perhaps null) of distinct universal quantifiers.
A query has the form $\exists A$ where $A$ is an unquantified query and $\exists$ is a string (perhaps null) of distinct existential quantifiers.

An unquantified assertion has the form $A_1 \land \ldots \land A_m$ where each $A_i$ is an elementary assertion.

An unquantified query has the form $A_1 \land \ldots \land A_m$ where each $A_i$ is an elementary query.

An elementary assertion is either an atom or a formula of the form $A \rightarrow p$ where $p$ is an atom and $A$ is an unquantified query.

An elementary query is either an atom, or a formula of the form $A \lor B$ where $A$ and $B$ are unquantified queries, or a formula of the form $A \rightarrow B$ where $B$ is an unquantified query and $A$ is an assertion (NB - not necessarily an unquantified assertion).

An atom is either an atomic sentence $f(t_1 \ldots t_n)$ (i.e., an n-ary predicate symbol followed by n terms), or its negation, $\neg f(t_1 \ldots t_n)$. This is the only context in which negation may appear. Additionally, an atom may be the propositional constant $t$ (but $\neg t$ is not allowed); $t$ corresponds to the ‘Id’ of the last two Chapters, the conjunction of all instances of Identity.

So for example $(\forall x)(f(x) \land g(x)) \rightarrow h(y)$ is an elementary query and by rule 5, $(\forall x)(f(x) \land g(x)) \rightarrow h(y) \rightarrow t(z)$ is an elementary assertion.

Note the use of the conjunction symbol ‘$\land$’ instead of the ‘&’ and ‘$\circ$’ of the last two chapters.

With this syntax we retain the normal PROLOG assertion restrictions: $A \lor B$ cannot be an assertion, and for an assertion $A \rightarrow B$, $B$ must be an atom (other than $t$), and $A$ must have the form of a query. The syntax extends PROLOG by allowing queries of the form $(\forall x_1) \ldots (\forall x_n)A \rightarrow B$, where $A$ is an unquantified assertion, and $B$ is a query (note that this is also an extension of the [GR84] syntax, which does not allow internal quantifiers). We discuss later on the expressive power of this language.
Where an assertion or query $A$ is a conjunction $A_1 \land \ldots \land A_m$ of elementary elements we will often abuse notation and write ‘$A_i \in A$’ to indicate that $A_i$ appears as a conjunct in $A$.

### 4.2.2 Substitutions

We will borrow all the standard terminology on substitutions; see for example [Rob79] or [Llo84]. A substitution is a set $\{s_1/x_1 \ldots s_n/x_n\}$ where $x_1 \ldots x_n$ are distinct variables and $s_1 \ldots s_n$ are terms, s.t. $x_i$ does not occur in $s_i$, for $i = 1 \ldots n$.

We will use Greek letters $\theta, \rho, \sigma, \tau \ldots$ to denote substitutions when we are not interested in their extensions. Where $A$ is a sentence, $A\theta$ represents the result of replacing for the free variables in $A$ as specified by $\theta$. We use $\theta\rho$ to denote the composition of $\theta$ and $\rho$, and we take the normal definition for most general unifier (mgu). We will use $\epsilon = \emptyset$ to denote the empty substitution. Some further details on substitutions may be found in Sec. 4.5.2.

### 4.2.3 Goal of the Computation

Given a database represented by an unquantified assertion $A$ and an unquantified query $G$ the goal is to determine substitutions $\theta$ such that $G\theta$ is an element of the smallest logic based theory containing $\forall A$ (we leave aside for the moment the question of which logic). As noted above, the atom $t$ here corresponds to the ‘I’d’ of our theoretical discussion: if $t$ is some conjunct of $A$, then the theory in question will be a regular theory.

### 4.2.4 The General Model

We will use the term ‘SLC’ (‘Linear condensed detachment for Conditional clauses with Selection rule’) for this model, by analogy with ‘SLD’. Note that we do not think of the ‘L’ as standing for ‘Linear resolution’ as it does in ‘SLD’; the basic operation is not Disjunctive Syllogism ($A, \neg A \lor B \vdash B$ - relevantly invalid) plus Unification but rather Detachment ($A, A \rightarrow B \vdash B$) plus Unification (ie,
We define an *SLC-proof* by elaborating on the standard definition of an SLD-refutation ([Llo84], p36). In addition to the more substantial extensions necessary for the proper handling of relevant implications, we will also make some minor changes to the way in which substitutions are handled. For completeness, we will include disjunctive goals in this formulation. The principle extensions to SLD resolution are as follows:

- A segmented database. New segments are generated in the course of the computation by implicational goals $A \rightarrow B$. The goal is reduced by forming a new database segment from $A$ and rewriting the goal by its consequent $B$. There will always be a ‘top-level’ segment, corresponding to the normal PROLOG database.

- A mechanism for keeping track of *usage*. The essential feature of the relevant system is that a goal $A \rightarrow B$ should succeed only if $B$ succeeds in a way which uses $A$. Thus we need to record which segments of the database have been used in the computation.

- Selective renaming of variables in rules. Consider the goal $f(x) \rightarrow g(x)$. We reduce this by adding $f(x)$ to the database and then reducing $g(x)$ (in a way which uses the added $f(x)$). Clearly any binding which is found for the one occurrence of $x$ must apply equally to the other; for example, if the database contains just $f(a) \rightarrow g(b)$ we would expect the query to fail, because the binding $\{b/x\}$ found in unifying $g(x)$ with the consequent of this rule must apply equally to the added $f(x)$, and $f(a)$ will not unify with $f(b)$. In this situation the occurrence of $x$ in the added $f(x)$ is to be taken as *free*, and so not renamed on invocation; we must also extend the SLD picture so that the answer substitution computed thus far is applied to the candidate selected from the database before attempted unification with the current goal. On the other hand, the syntax allows queries of the form $(\forall x)f(x) \rightarrow g(y)$, and in this case the added element has an explicit universal quantifier, so the $x$ *will* be renamed when $f(x)$ is selected to
match the current goal. In the 'top-level' segment, all assertions are taken
to be universally quantified, and so their variables are always renamed on
selection; applying the answer substitution to these candidates will not
change them, since it will not mention their newly-created variables.

- Negated atoms. Two atoms $p$ and $q$ unify only if either both or neither have
  a leading ‘~’. For example, $\neg f(a)$ and $\neg f(x)$ will unify with mgu $\{a/x\}$,
  but $\neg f(a)$ and $f(x)$ do not unify. For all purpose, $f(a)$ and $\neg f(a)$ are just
distinct atoms.

Formally, the structures and operations are as follows:

- An indexed elementary query has the form $A^n$ where $A$ is an elementary
  query and $n$ is a positive integer. The index will determine which segments
  of the database may act as sources for matching rules. Where $A$ is a con­
  junction of elementary queries $A_1 \land \ldots \land A_m$ we will use the notation $[A]^n$
to stand for the conjunction of indexed elementary queries $A_1^n \land \ldots \land A_m^n$.

- A computation rule $R$ is a mapping from conjunctions of indexed elementary
  queries to indexed elementary queries, such that $R(A_1 \land \ldots \land A_n) = A_i$ for
  some $1 \leq i \leq n$.

- A database is a tree (with a single root) of database segments $S_1 \ldots S_n$. Each
  segment $S_i$ is an assertion of the form $\forall_i A_i$ where $A_i$ is an unquantified
  assertion and $\forall_i$ is a string of distinct universal quantifiers (perhaps null,
  and not necessarily binding every free variable in $A_i$).

- Where $Q$ is a string of distinct universal quantifiers $(\forall x_1) \ldots (\forall x_n)$, a $Q$-
  renaming is a substitution $\{x_1'/x_1 \ldots x_n'/x_n\}$ where $x_1' \ldots x_n'$ are some dis­
  tinct new variables for this renaming, from a stock of variables set aside for
  this purpose.

- Where $S_i$ is a segment in a database $D$, we let $\text{Acc}(i, D)$ be the set of
  segments consisting of $S_i$ and all its ancestors in the database tree, up to
  and including the root of the tree. This is the set of segments accessible to
  $S_i$. 
A simple way to view an implicational goal \( A \rightarrow B \) is as an instruction to set up a subproof with assumptions \( A \) and subgoal \( B \), with the stipulation that \( A \) must actually be used in the reduction of \( B \). In the general case, with no restriction on the computation rule, the computation path is allowed to jump around from subproof to subproof in an arbitrary fashion. The tree structure of the database reflects the nesting of subproofs. Each goal is contained within one of these subproofs, and candidates for matching it may come from the assumptions of its own subproof or any of those which recursively contain it: in terms of the database tree \( D \), to match an indexed goal \( A^i \) we may select candidates from any segment in \( Acc(i, D) \). If we take a fixed left-to-right computation rule, as in the computation tree model of the next section, the picture is somewhat simpler (and probably more natural). In this case, the subproof generated by the goal \( A \rightarrow B \) must be completed before proceeding to the next goal at the same level, and so instead of carrying all the segments generated through the entire proof we can maintain a stack, popping it at the conclusion of each subproof (‘discharging the assumptions’). There is only ever one ‘active’ subproof and hence no need for indices on goals; all the segments in the stack are accessible at each step.

**Definition 7** An SLC state \( \sigma \) is a quadruple

\[
\langle G, D, U, \theta \rangle
\]

where:

- \( G \) is a conjunction (possibly empty) of indexed elementary queries.
- \( D \) is a database.
- \( U \subseteq \{1 \ldots n\} \) (recording segment usage).
- \( \theta \) is a substitution.

**Definition 8** Let \( \sigma_1 \) and \( \sigma_2 \) be SLC states, let \( G_1 = A_1 \land \ldots \land A_m \), let the segments in \( D_1 \) be \( S_1 \ldots S_n \) and let \( R \) be a computation rule such that \( R(G_1) = A_i \). Then \( \sigma_2 \) is derived from \( \sigma_1 \) via \( R \) if one of these conditions is satisfied, depending on \( A_i \):
• $A_i$ is a disjunction $(B \lor C)^k$. Then $\sigma_2$ is the same as $\sigma_1$ except that $G_2 = A_1 \land \ldots A_{i-1} \land [D]^k \land A_{i+1} \ldots \land A_m$ where $D$ is either $B$ or $C$.

• $A_i$ is an implication $(B \rightarrow C)^k$. Then $\sigma_2$ is the same as $\sigma_1$ except that $D_2$ includes the new segment $S_{n+1} = B$ (which has the form of a database segment, by the syntax for implicational queries) added to the database tree as a child of $S_k$ (that is, the segment indicated by the index on $A_i$), and $G_2 = A_1 \land \ldots A_{i-1} \land [C]^{n+1} \land A_{i+1} \ldots \land A_m$. Additionally, if $t \in B$, $U_2 = U_1 \cup \{n + 1\}$. Note that $S_k$, and whatever is accessible to it, is accessible to the new $S_{n+1}$.

• $A_i$ is an atom $p^k$. Then there is an $S_i = \forall_i C_i \in Acc(k, D_1)$ such that for some $B \rightarrow p' \in C_i$,
  
  $- \rho$ is a $\forall_i$-renaming.
  
  $- p\theta_1$ and $p'\rho\theta_1$ unify with mgu $\theta$
  
  $- \theta_2 = \theta_1\theta$
  
  $- G_2 = A_1 \land \ldots A_{i-1} \land [B\rho]^k \land A_{i+1} \land \ldots \land A_n$.
  
  $- U_2 = U_1 \cup \{i\}$
  
  $- D_2 = D_1$.

Of course, the matching element from the database may be an atom instead of an implication, in which case $A_i$ is effectively deleted from $G_2$, rather than rewritten.

No states are derivable from a state with the empty goal.

Note the way that in the reduction of an implicational goal, if the new segment contains $t$ as a conjunct then it is automatically marked as used. Remember that $t$ is to be thought of as the conjunction of all instances of Identity, and for any goal $G$, $G \rightarrow G$ can always be ‘used’ in a trivial way in the reduction of $G$, i.e., by rewriting it with a new copy of itself.

There is a slight change in the handling of substitutions compared to the SLD model of [Llo84]: instead of the substitution attached to each state being the
mgu generated by the unification involved in the derivation of the state, it is instead the complete ‘answer substitution’ generated so far. This is an inessential modification but it facilitates the development. Note also that the substitution must be applied to the rule selected from the database; this serves to establish the linkage between antecedent and consequent variables discussed above.

**Definition 9** Let \( \sigma \) be an SLC state, and let \( R \) be a computation rule. An SLC-derivation from \( \sigma \) via \( R \) is a (finite or infinite) sequence of SLC states \( \sigma_1, \sigma_2 \ldots \) such that:

- \( \sigma_1 = \sigma \)
- \( \sigma_{n+1} \) is derived from \( \sigma_n \) via \( R \).

A derivation may be either finite or infinite. A successful derivation is a finite derivation whose last state \( \sigma_k \) has \( G_k \) as the empty goal, and \( U_k = \{1 \ldots n\} \), where \( S_1 \ldots S_n \) are all the segments in \( D_k \), indicating that all segments have been used. A failed derivation is a finite derivation where nothing is derivable from the last state.

**Definition 10** The initial state for an unquantified query \( G \) and an unquantified assertion \( A \) is the SLC state

\[
([G]^1, D_1, U_1, \epsilon)
\]

\( D_1 \) contains a single segment \( \forall_1 A \) where \( \forall_1 \) contains a universal quantifier for every free variable in \( A \). \( U_1 \) is either \( \{1\} \) or \( \emptyset \), depending on whether or not \( t \in A \).

Note the way that the ‘top-level’ segment is automatically marked as used if it contains \( t \); in other words, if it corresponds to a regular theory. See the comment above on the rule for reducing implicational goals.

**Definition 11** An SLC-proof with answer \( \theta \) of a query \( G \) from an assertion \( A \) via \( R \) is a successful derivation from the initial state for \( G \) and \( A \) via \( R \) with last state \( \sigma_k \) such that \( \theta = \theta_k \).
The space of SLC computations for a particular $G$, $D$ and $R$ is represented by an SLC-tree (cf [Llo84], p48).

**Definition 12** Let $A$ be an unquantified assertion, $G$ an unquantified query and $R$ a computation rule. Then the SLC-tree for $G$ from $A$ via $R$ is defined as follows:

- Each node of the tree is an SLC state.
- The root node is the initial state for $G$ and $A$.
- Each node has as its descendants all the SLC states derivable from it.

Each branch of the SLC-tree is an SLC derivation; either a proof, or a failed derivation, or an infinite derivation.

SLC involves quite minor extensions to the SLD model. If we restrict ourselves to normal PROLOG syntax we get SLD resolution (modulo the minor differences in substitution handling, and the indices which are redundant in this case), since the database has at all times only a single segment with all variables universally quantified, and there are no implicational goals. In these restricted circumstances, when we apply the answer substitution to the head of the matching rule in the reduction of an atomic goal, all free variables in the (renamed) candidate are new and hence the substitution has no effect.

Of the new elements, the method of checking usage is quite straightforward. A subproof can only succeed if its assumptions are actually used (which is what makes the system 'relevant') and so a proof as a whole can only succeed if the assumptions of each of its subproofs have been used; in other words, for a derivation to be successful it must have reduced the initial goal in a way which uses each segment. Note that the candidate selection process is constructed in such a way that the assumptions of a subproof may only be used within that subproof, or subproofs contained in it. The question of what logic will allow us to get away with this straightforward scheme is more interesting than might appear at first sight; perhaps surprisingly, the main issue will be the precise interpretation of our conjunction symbol ‘$\land$’.
Theorem 67 (Independence of the computation rule) If there is an SLD-proof of \( G \) from \( D \) via \( R \) with answer \( \theta \) then for any computation rule \( R' \), there is a proof of \( G \) from \( D \) via \( R' \) with answer \( \theta' \), where \( G\theta \) and \( G\theta' \) differ at most by being variants.

The proof of this theorem follows the lines of the corresponding result for SLD-refutations ([Llo84], Th. 9.2), with the obvious complications. We can use this result to generate the computation tree model of the next section in which the handling of usage is more straightforward, and which has a strong resemblance to the Gentzen formulations of later sections.

4.2.5 The Computation Tree Model

Here we present a model which is an extension of the computation trees of [GR84], deductively equivalent to SLC proofs with a left-to-right computation rule.

A computation tree is a finite tree with a single root. Each node has the form

\[ ((S_1 \ldots S_m), G, U_1, U_O, \theta_I, \theta_O) \]

where the \( S_i \) are database segments (as for the SLC model), \( G \) is an unquantified query, \( \theta_I \) and \( \theta_O \) are substitutions and \( \emptyset \subseteq U_I \subseteq U_O \subseteq \{1 \ldots m\} \). The sets \( U_I \) and \( U_O \) record which of the database segments \( S_1 \ldots S_m \) have been used in the computation, with \( U_I \) recording those segments used in computation previous to the subtree rooted at the node, and \( U_O \) adding the segments used in that subtree (where we read the order of computation in a depth-first, left-to-right manner). Similarly, \( \theta_I \) and \( \theta_O \) are the answer substitutions generated by the computation, divided into ‘input’ and ‘output’ in the same way.

The nodes which may appear in a tree are constrained as follows:

**Tip** A tip node has the form

\[ ((S_1 \ldots S_m), p, U, U \cup \{i\}, \theta_I, \theta_I\theta) \]
where:

- \( p \) is an atom.
- \( S_i = \forall_i A_i \)
- \( p' \in A_i \).
- \( \rho \) is a \( \forall_i \) renaming, and \( p\theta_1 \) and \( p'/\rho\theta_1 \) unify with mgu \( \theta \).

All other nodes obey the following rules, depending on the form of the query:

**Fuse** If the node has the form

\[
((S_1 \ldots S_m), A \land B, U_1, U_0, \theta_1, \theta_0)
\]

then it has exactly two children:

\[
((S_1 \ldots S_m), A, U_1, U_0, \theta_1, \theta_0)
\]

and

\[
((S_1 \ldots S_m), B, U_1, U_0, \theta_1, \theta_0)
\]

for some \( U \) and \( \theta \).

**Vel** If the node has the form

\[
((S_1 \ldots S_m), A \lor B, U_1, U_0, \theta_1, \theta_0)
\]

then it has a single child:

\[
((S_1 \ldots S_m), C, U_1, U_0, \theta_1, \theta_0)
\]

where \( C = A \) or \( C = B \).

**Chain** If the node has the form

\[
((S_1 \ldots S_m), p, U_1, U_0, \theta_1, \theta_0)
\]

where \( p \) is an atom then the node has exactly one child:
\[(\langle S_1 \ldots S_m \rangle, B_\rho, U_I, U, \theta_I, \theta_O)\]

where:

- Some \( S_i \) is \( \forall_i A_i \).
- \( B \rightarrow p' \in A_i \).
- \( \rho \) is a \( \forall_i \)-renaming, and \( p^\theta_I \) and \( p'^\rho \theta_I \) unify with mgu \( \theta \).
- \( U_O = U \cup \{i\} \).

\textbf{Sub} If the node has the form

\[(\langle S_1 \ldots S_m \rangle, A \rightarrow B, U_I, U_O, \theta_I, \theta_O)\]

then it has exactly one child:

\[(\langle S_1 \ldots S_m, A \rangle, B, U'_I, U, \theta_I, \theta_O)\]

where:

- \( m + 1 \in U \)
- \( U_O = U - \{m + 1\} \)
- \( U'_I = U_I \cup \{m + 1\} \) if \( t \in A \) or otherwise \( U'_I = U_I \).

Where \( \kappa \) is a computation tree node we will use the notation \textbf{Proof} \((\kappa)\) to indicate that the existence of a computation tree with root \( \kappa \). Given an unquantified assertion \( A \) representing the database and an unquantified query \( G \) the goal of the computation is to find substitutions \( \theta \) for which

\textbf{Proof} \((\forall A), G, U, \{1\}, \epsilon, \theta)\)

is true, where \( U \) is either \( \{1\} \) or \( \emptyset \), depending on whether or not \( t \in A \).

To illustrate the computation we will construct a computation tree based on a simplified form of the planning example of Chapter 1.

Let the database consist of just these elements:
CHAPTER 4. RELEVANT LOGIC PROGRAMMING

r₁ \text{ state(zone2, factory)} \rightarrow \text{state(zone1, polluted)}

r₂ (\text{state}(x, s) \rightarrow \text{state}(y, t)) \land \text{bad-state}(y, t) \rightarrow \text{bad-state}(x, s)

f₁ \text{ state(zone3, danger-to-pop)}

f₂ \text{ bad-state}(x, polluted)

f₃ \text{ bad-state}(x, danger-to-pop)

Let \( S₁ = \forall(r₁ \land r₂ \land f₁ \land f₂ \land f₃) \) and let us ask the query \( G₁ = \text{bad-state(zone2, w)} \) ('What would be a bad state for zone2?').

Step 1 The root node is

\[ ((S₁), G₁, \emptyset, U₁, \epsilon, \theta₁) \]

where \( U₁ \) and \( \theta₁ \) ('the answer') are to be determined. For a successful computation, \( U₁ \) must end up being the singleton set \{1\}.

Step 2 Using the Chain rule and \( r₂ \) we generate the root's child:

\[ ((S₁), G₂, \emptyset, U₂, \theta₂, \theta₁) \]

Rule \( r₂ \) is contained in the first (and so far only) segment which is universally closed, so before unifying we apply the renaming \( \rho = \{x₁/x, s₁/s, y₁/y, t₁/t\} \).

\( G₂ \) is then the body of \( r₂\rho \):

\[ \text{state}(x₁, s₁) \rightarrow \text{state}(y₁, t₁) \land \text{bad-state}(y₁, t₁) \]

\( G₁ \) and \( \text{bad-state}(x₁, s₁) \) (the head of \( r₂\rho \)) unify with mgu \( \theta = \{\text{zone2/x₁, w/s₁}\} \), and \( \theta₂ = \epsilon\theta = \theta \). Finally, \( U₁ = U₂ \cup \{1\} \).

Step 3 The new goal \( G₂ \) is conjunctive so the tree now branches in accordance with the Fuse rule. On the left we have

\[ ((S₁), \text{state}(x₁, s₁) \rightarrow \text{state}(y₁, t₁), \emptyset, U₃, \theta₂, \theta₃) \]

and on the right
where $U_3$ and $\theta_3$ are to be determined; the values given as output by the left branch are fed to the right branch as input. Taking the left branch first:

**Step 4** The goal is implicational so its child is determined by the Sub rule, giving:

$$((S_1, S_2), \text{state}(y_1, t_1), \emptyset, U_4, \theta_2, \theta_3)$$

where $U_4$ is to be determined, with the constraint that $U_3 = U_4 - \{2\}$ and $2 \in U_4$ (to indicate that the new segment has been used in the computation of the new goal). $S_2 = \text{state}(x_1, s_1)$; note that $x_1$ and $s_1$ are free here. Since $S_2$ does not contain $t$, the 'input' use record remains the same as for this node's parent; i.e., the empty set.

**Step 5** The new goal matches with the head of $r_1$ so we use the Chain rule again:

$$((S_1, S_2), \text{state}(y_1, t_1), \emptyset, U_5, \theta_4, \theta_3)$$

We computed $\theta_2$ at Step 2: it does not affect any of the variables in $\text{state}(y_1, t_1)$. This and the head of $r_1$ unify with $\text{mgu} \theta = \{\text{zone1}/y_1, \text{polluted}/t_1\}$. $\theta_4 = \theta_2 \theta = \theta_2 \cup \theta$. Finally, $U_4 = U_5 \cup \{1\}$.

**Step 6** We can now form a Tip node, matching the goal with the single atom contained in $S_2$, $\text{state}(x_1, s_1)$. Because all the variables in this segment are free, no renaming takes place. We have $\text{state}(x_1, s_1)\theta_4 = \text{state}(\text{zone2}, w)$ and this unifies with the goal $\text{state}(\text{zone2}, \text{factory})$ with $\text{mgu} \theta = \{\text{factory}/w\}$. In accordance with the Tip rule we set $U_5 = \emptyset \cup \{2\} = \{2\}$. $\theta_3$ becomes

$$\theta_4\theta = \{\text{zone2}/x_1, \text{factory}/s_1, \text{zone1}/y_1, \text{polluted}/t_1, \text{factory}/w\}$$

This concludes the left branch of the fork at Step 3. We move now to the right branch.

**Step 7** We can form another Tip node, matching the goal with $f_2$. $\text{bad-state}(y_1, t_1)\theta_3 =$ $\text{bad-state}(\text{zone1}, \text{polluted})$. Renaming $f_2$ with $\{z_1/z\}$ gives an $\text{mgu}$ of $\theta = \{\text{zone1}/z_1\}$ and we have $\theta_1 = \theta_3 \theta = \theta_3 \cup \theta$. Finally, $U_2 = U_3 \cup \{1\}$.
The computed answer, \( \theta_1 \), gives

\[
G_1 \theta_1 = \text{bad-state}(\text{zone2},w) \theta_1 = \text{bad-state}(\text{zone2},\text{factory})
\]

We still need to check that all the constraints satisfied:

- \( U_5 = \{2\} \) (Step 6).
- \( U_4 = U_5 \cup \{1\} = \{1,2\} \) (Step 5).
- \( 2 \in U_4 \), as required, and \( U_3 = U_4 - \{2\} = \{1\} \) (Step 4).
- \( U_2 = U_3 \cup \{1\} = \{1\} \) (Step 7).
- \( U_1 = U_2 \cup \{1\} = \{1\} \), as required (Steps 2 and 1).

If we compare our computation trees to those of [GR84], we see that the only essential differences are the segmented database and the mechanism for keeping track of usage. If we set things up so \( t \) occurs in the top-level segment, and so that \( t \) occurs as a conjunct in \( A \) for every implicational query \( A \rightarrow B \), then every segment created in the course of the computation will be automatically marked as used (cf the Sub rule), and the usage mechanism and segmented database will become redundant. This is the computational expression of the fact that for the positive fragments of the two logics, relevant \( A \land t \rightarrow B \) is equivalent to intuitionistic \( A \supset B \) ([Mey72]). So we can get the intuitionistic NPROLOG of [GR84] from this scheme in one of two ways: either by replacing every implicational query \( A \rightarrow B \) by \( A \land t \rightarrow B \) (and making sure that \( t \) is in the initial database segment), or by ‘turning off’ all use requirements.

To illustrate our contention (from Chapter 1) that the ‘bad-state’ predicate given here will not work for an irrelevant system like NPROLOG, consider Step 5 of the above example. At this point we could have matched with f1, forming a Tip node with \( \theta_3 = \{\text{zone2}/x_1, w/s_1, \text{zone3}/y_1, \text{danger-to-pop}/t_1\} \). This would set \( U_4 \) to \( \{1\} \). Making the same move for the right hand branch as we did before at Step 7, we can see that the final answer substitution will leave the query variable \( w \) unbound, giving the answer ‘Everything’. In the relevant system, this
Chapter 4. Relevant Logic Programming

Computation path is blocked by the constraint at Step 4, \( 2 \in U_4 \); in other words, by the requirement that \( S_2 \), the antecedent of the conditional query, actually be used.

4.2.6 Comparing SLC Proofs and Computation Trees

Here we show that computation trees are deductively equivalent to successful SLC derivations with a computation rule which always selects the left-most element of a goal; that is, the rule \( R_L \) where \( R_L(G_1 \land \ldots G_n) \) is always \( G_1 \).

Theorem 68 There is an SLC proof with answer \( \theta \) of \( G \) from \( A \) via \( R_L \) iff \( \text{Proof}((\forall A),G,D,\{1\},e,\theta) \), where \( D \) is either \( \{1\} \) or \( \emptyset \), depending on whether or not \( t \in A \).

Proof

- Left to right. We sketch how indices on goals and the tree structure of the database can be made redundant in the SLC derivation. Consider a state in the derivation, \( \sigma_j \), where the left-most element of the goal at this step is an implication \( (B \rightarrow C)^j \), and the database contains segments \( S_1 \ldots S_n \). At \( \sigma_{j+1} \) the left most element of the goal is \( [C]^{n+1} \) and the database has a new segment \( S_{n+1} = B \) added as a child of \( S_1 \). With the computation rule \( R_L \), the next sequence of states clearly acts to reduce \( [C]^j \) to the empty goal; all the action takes place within the 'subproof' set up with \( S_{n+1} = B \) as its assumptions. Say that this process is completed at stage \( \sigma_{j+m} \).

Clearly, \( S_{n+1} \) will only be accessible in the sequence \( \sigma_{j+1} \ldots \sigma_{j+m} \), and for a successful derivation overall, it must be the case that \( n + 1 \in U_{j+m} \); if \( S_{n+1} \) has not been used at that point, it never will be. It is easy to see, then, that \( S_{n+1} \) may safely be removed from the database after \( \sigma_{j+m} \). If we recursively apply this operation to all 'sub-proofs' in the derivation, we see that at any stage, the database has a single branch (in other words, a chain), all of whose segments are accessible. So the indices on goals are no longer required. Equally, after \( \sigma_{m+1} \) we can remove the '\( n+1 \)' from the use.
record, and applying this operation recursively, we end up with a last state $\sigma_k$ in which $U_k = \{1\}$. The remainder of the proof consists of simple moves converting the linear SLC proof into an AND tree, and attaching \textquote{input'} and \textquote{output'} usage records and substitutions as appropriate.

• Right to left. Details left to the reader.

With Th. 67 this gives the following statement of the general equivalence of the two models:

\textbf{Theorem 69} There is an SLC proof of $G$ from $A$ with answer $\theta$ (via some computation rule $R$) if and only if $\text{Proof}(\forall A), G, D, \{1\}, e, \theta'\rangle$ where $D$ is either $\{1\}$ or $\emptyset$, depending on whether or not $t \in A$, and $\theta'$ differs from $\theta$ at most in $G\theta$ and $G\theta'$ being variants.

We will generally use the computation tree model in the discussions to follow, but as this theorem shows, all results apply equally to SLC proofs.

\section*{4.2.7 A Conditional Query Evaluation Procedure}

CQEP is a fairly straightforward extension of the standard PROLOG evaluation procedure. We present it here in a form very close to the CLOGPROG implementation, described briefly later in the paper. CQEP restricts the computation tree model by introducing deterministic candidate selection (retaining the left-to-right computation rule).

A \textit{computation state} is a quintuple

$$(\text{Goals, Segments, } \theta, \text{Cand, Used})$$

where:

• \textbf{Goals} is a stack of elements which are either unquantified queries or positive integers (which function as check points for usage in a way to be described
shortly). The top of this stack is the current goal to be reduced, and the computation terminates when this stack has been reduced to the empty state, indicating that all goals have been reduced.

- **Segments** is an indexed stack of *database segments*. These have the indices range from 1 for the bottom of the stack to \(n\) for the top of an \(n\)-element stack. The first element on the stack (i.e., the element with index 1) is the 'base-level' database; for normal PROLOG purposes, the stack will never grow beyond this. Additional elements represent the antecedents of conditional goals generated in the course of the computation.

- **\(\theta\)** is the answer substitution generated to this point.

- **Cand** is a pointer to an elementary assertion in **Segments**, the next candidate for matching with the current goal. We think of this as a pair \((i, j)\); where the \(i\)-th element on the segment stack is \(\forall_i(A_1 \land \ldots \land A_j \land \ldots \land A_m)\), \((i, j)\) points to the elementary assertion \(A_j\).

- **Used** is a subset of \(\{1 \ldots n\}\) where the index of \(\text{Top}(\text{Segments})\) is \(n\). It records which of the database segments have been used in the computation to this point.

The *computation stack* \(C\) is a stack of computation states, maintained for backtracking.

**The Procedure**

The evaluation procedure is called with an assertion \(A\) representing the 'base-level' database and a query \(G\). It returns either 'FAIL' or an answer substitution \(\theta\). The top level routine is as follows:

- `New(C)`
- `New(Goals)`
- `New(Segments)`
- `Cand = NIL`
\[ \theta = \epsilon \]

Used = \emptyset

* Push a check-point for the 'top-level' segment *

Push-Goal(1,\texttt{Goals})

Push-Goal(G,\texttt{Goals})

Push-Segment(\forall A)

While Not (Empty(\texttt{Goals})) Do

Reduce-Goals

Return(\theta)

The Reduce-Goals procedure:

Case Top(\texttt{Goals}) of

\[ p: \text{Reduce()} \quad * \text{An atomic goal} * \]

\[ A \land B : \text{Pop(\texttt{Goals})} \]

* Left-to-right, depth-first evaluation *

Push-Goal(B,\texttt{Goals})

Push-Goal(A,\texttt{Goals})

\[ A \lor B: \text{Pop(\texttt{Goals})} \]

* Try the first disjunct, having set up a backtrack *

* point to try the second if it fails. *

Copy(\texttt{Goals},\texttt{Goals}_1)

Push-Goal(A,\texttt{Goals})

Push-Goal(B,\texttt{Goals}_1)

Cand = \textsc{Nil}

Push-State((\texttt{Goals}_1, \texttt{Segments}, \theta, \texttt{Cand}, \texttt{Used}))

\[ A \rightarrow B: \text{Pop(\texttt{Goals})} \]

* The antecedent becomes the top segment, *

* the consequent the new goal, and we need *

* to check that \( A \) is actually used in the *

* reduction of \( B \). *

Push-Segment(A)
Push-Goal(Index(Top(Segments)), Goals)
Push-Goal(B, Goals)

\( N: \)
* The use check.
* \( N \) will always be the index of the top of \( \text{Segments} \).
* The check succeeds if this segment has * been used.

\[ \text{If } N \in \text{Used Then} \]
\[ \text{Used} = \text{Used} - \{N\} \]
\[ \text{Pop}(\text{Segments}) \quad * \text{Discharge the antecedent} * \]
\[ \text{Pop}(\text{Goals}) \]

\[ \text{Else} \]
\[ \text{Backtrack} \]

---

The Reduce procedure:

* \( \text{Top(Goals)} \) is an atomic query \( p \).
* If this is the first attempt at reducing \( p \), select the first * candidate for matching it. Otherwise, we have reached *
* this point by backtracking and \( \text{Cand} \) will point to the *
* next candidate to try *

\[ \text{If } \text{Cand} = \text{NULL Then} \quad \text{Cand} = \text{Select-First-Cand}(p) \]
* Backtrack if no candidates found *

\[ \text{If } \text{Cand} = \text{NULL Then} \]
\[ \text{Backtrack} \]

\[ \text{Else} \]
* Set up a backtrack point if there are further possibilities *
\[ \text{Cand}_1 = \text{Select-Next-Cand}(p, \text{Cand}) \]

\[ \text{If } \text{Cand}_1 \neq \text{NULL Then} \]
\[ \text{Push-State}((\text{Goals, Segments, } \theta, \text{Cand}_1, \text{Used})) \]
* Extract the candidate, renaming quantified variables, and *
* record the index of the segment it came from. *
Try = Rename(Cand)
Try-Segment = Segment(Cand)
If Unify(p, Head(Try), θ, θ') Then
* Record the fact that this segment has been used, and *
* replace the current goal with the body of the candidate. *
θ = θ'
Used = Used ∪ {Try-Segment}
Pop(Goals)
Push-Goal(Body(Try), Goals)
Cand = NIL
Else
Backtrack

- Unify(p, p', θ, θ') returns 'true' if pθ and p'θ unifiy, and 'false' otherwise. Where θ'' is the mgu, θ' is set to θθ''.
- The Backtrack procedure retrieves the most recently saved computation state on the computation stack C, and pops the stack; or else exits from the entire computation with a 'FAIL' result if there are no more choice points.
- The Push-Goal procedure pushes a query onto the goal stack given as argument.
- The Push-Segment procedure takes an assertion \( S_i = \forall_i A_i \) (which by the syntax for assertions has the form of a database segment) and pushes it onto Segments. If \( t \in A_i \), the index of the new top of Segments is added to Used.
- Segment(Cand) returns the segment index component of the Cand pointer.
- The Rename procedure takes a pointer to an elementary assertion B in a segment \( \forall_i A_i \) on the segment stack, constructs a \( \forall_i \)-renaming \( ρ \), and returns \( Bρ \).
- Push-State pushes a new computation state onto the computation stack, setting up a backtrack point.
CHAPTER 4. RELEVANT LOGIC PROGRAMMING

For the case where the elementary assertions in the 'base-level' segment, the normal PROLOG database, are either facts or first-degree implications (ie, no nested implications), this is just a model of the normal PROLOG scheme if we impose a left-to-right ordering on candidate selection for the base-level. The segment stack does not change in the course of the computation and the Rename procedure will always produce candidates all of whose variables are new since the 'base-level' segment is taken as universally closed. So we get normal PROLOG behaviour for normal PROLOG operations.

The only point remaining open is the ordering on elements in the segmented database used by Select-First-Cand and Select-Next-Cand. In the CLOGPROG implementation we generalize the PROLOG scheme within segments; we always consider $A_i$ before $A_{i+1}$ in a segment $\forall_j(A_1 \ldots A_m)$. The question then is how to order the segments themselves. The simplest approach is just to start with the top of the segment stack and move down it sequentially until we finally reach the base level, and this is probably fine for most purposes where we would not expect the segment stack to have a size of more than two or three. However, it seems likely that the termination behaviour of the computation will be improved if we try to ensure maximum coverage of the stack; note that at any point in the computation we know that every element in the segment stack at that point must be used somehow if the computation is to succeed; the final state of any successful computation has just the base level remaining. So our preferred selection strategy is to first of all search those segments whose indices have not been recorded in Used, and then those that have. Precisely:

- Let $n$ be the index of Top(Segments).
- Let $(i_1 \ldots i_m)$ be the sequence formed by placing the elements of $\{1 \ldots n\} - \text{Used}$ in descending order.
- Let $(i_{m+1} \ldots i_n)$ be the elements of Used in descending order.
- The order in which segments are searched (in Select-First-Cand and Select-Next-Cand) is then given by $(i_1 \ldots i_n)$. Note that this ordering is completely determined by the current state of the computation, specifically by
Segments and Used.

To illustrate the issues and the evaluation procedure generally, consider a proof of $(A \land B \rightarrow B) \rightarrow A \rightarrow A \rightarrow B \rightarrow B$ from the database $S_1$ containing just $t$. We will denote stacks by $[E_1 \ldots E_n]$ where $E_n$ is the top of the stack.

The initial state of the evaluation has $\text{Segments} = [S_1]$, $\text{Goals} = [1, (A \land B \rightarrow B) \rightarrow A \rightarrow A \rightarrow B \rightarrow B]$ and $\text{Used} = \{1\}$ (the answer substitution will always be the empty substitution $\epsilon$). Note the way $\text{Used}$ gets set to $\{1\}$ as part of the initialization, since the base-level segment $S_1$ contains $t$. The first part of the evaluation involves successively pushing antecedents onto $\text{Segments}$ and check-points onto $\text{Goals}$, until we reach the point where $\text{Segments} = [t, A \land B \rightarrow B, A, A, B]$ and $\text{Goals} = [1, 2, 3, 4, 5, B]$. That is, $\text{Goals}$ has a check-point for each of the five segments created, and the top goal $B$. $\text{Used}$ has not changed, since none of the second to the fifth segments contains $t$. From this point the computation proceeds as follows:

1. We can match $B$ using either the second or the fifth (top) segments; we try the fifth one first, but this leads to a dead end, as check-point 4 will fail, so we backtrack and choose $A \land B \rightarrow B$ from the second segment.

2. The next state has $\text{Goals} = [1, 2, 3, 4, 5, B, A]$, $\text{Used} = \{1, 2\}$ and $\text{Segments}$ unchanged. The first match for the goal $A$ is the $A$ from segment 4, giving:

3. $\text{Goals} = [1, 2, 3, 4, 5, B]$ and $\text{Used} = \{1, 2, 4\}$. Since segment 5 has not been used we pick the $B$ there to match the goal $B$, but now check-point 3 will fail so we backtrack to select $A \land B \rightarrow B$ from segment 2 again:

4. $\text{Goals} = [1, 2, 3, 4, 5, B, A]$ and $\text{Used} = \{1, 2, 4\}$. If we were now to match the goal $A$ with the $A$ from segment 4 again, the resulting state would be identical to the one at 3, and the evaluation would loop. Since segment 4 has been used, however, we pass over it and take the $A$ from segment 3 instead:

5. $\text{Goals} = [1, 2, 3, 4, 5, B]$ and $\text{Used} = \{1, 2, 3, 4\}$. Now out of the loop, we make a third attempt at matching with the $B$ from segment 5:
6 Goals = \{1, 2, 3, 4, 5\} and Used = \{1, 2, 3, 4, 5\}. All that remain are the use-checks, which succeed since all segments have now been covered.

The formula we just proved has some small interest; under the translation we discuss later it is equivalent to the following:

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

which has something of the status as a test for relevant automated theorem proving systems as Peirce's Law for classical theorem provers, for example (see [TMM88, p.15]). It is nice to see our system handling it with reasonable efficiency.
4.3 The Logic

In this section we give a formal description of the fragment of relevant logic computed by these models. To illustrate that the logical questions are more involved than might appear at first sight, consider a database consisting of just \( t, A \rightarrow C \) and \( A \). Clearly the query \( B \rightarrow C \) will fail, whereas \( A \) and \( A \land B \rightarrow C \), and so \( A \land (A \land B \rightarrow C) \), will all succeed. But readers may satisfy themselves that \( A \land (A \land B \rightarrow C) \rightarrow B \rightarrow C \) will also succeed, demonstrating that the class of queries which succeed from a given database is not generally closed under Modus Ponens. The explanation of this peculiarity will hinge on an ambiguous translation of \( \land \) in terms of two different logical conjunctions.

4.3.1 The Logical Framework

Here we provide a bridge between the logical discussion of Chapters 2 and 3 and the detailed discussion of the fragment to follow. As stated previously in this chapter, the goal of the logic programming system, given an assertion \( A \) and a query \( G \), is to find substitutions \( \theta \) for which \( G\theta \) is in the smallest logic-based theory (cf Sec. 2.11) containing \( A \). Recall that a logic-based theory for a logic \( \mathcal{L}^\alpha \) is any set of sentences \( T \) closed under Adjunction and Detachment for \( \mathcal{L}^\alpha \) logical implication: \( B \in T \) whenever \( A \in T \), for all \( A \rightarrow B \in \mathcal{L}^\alpha \). The logic which will form the basis for our computation is \( \mathcal{Q}\mathcal{L}_{+}^{DTE} \), which we showed in Th. 41 to be the same as \( RQ_+ \), the positive fragment of the major relevant logic \( RQ \) (but see below for a hiatus in this story). We will show in this section that within this logic we can represent a substantial (if asymmetric) fragment of the full logic \( \mathcal{L}_{+}^{DTE} \) (which includes \( \neg, + \) and \( \rightarrow \) in its vocabulary), utilizing the results of Ths. 9 and 44.

From now on we will only be concerned with entailment systems which include the \( D, T \) and \( E \) elimination principles.

To generate logic-based theories, we introduce a new consequence operation, \( \mathcal{C}_T^\alpha \), where for any entailment system \( \mathcal{E}^\alpha \),
$$C^\alpha_L(A) = C^\alpha(A \circ Id^\alpha)$$

Recall from Sec. 2.10.2 that $B \in C^\alpha(A)$ iff $A \Rightarrow B$ is provable in $\mathcal{E}^\alpha$. So $B \in C^\alpha_L$ iff $A \circ Id^\alpha \Rightarrow B$ is provable $\mathcal{E}^\alpha$, and since $D \in \alpha$, this is the case just where $A \Rightarrow B \in \mathcal{L}^\alpha$ (Th. 29). It is not hard to see that $C^\alpha_L$, like $C^\alpha$, is a Tarski-style consequence operation, and that $C^\alpha_L(A)$ is indeed the smallest $\mathcal{L}^\alpha$ theory containing $A$.

We also introduce a consequence operation specifically for regular theories, $C^\alpha_R$, defined as $C^\alpha_R(A) = C^\alpha(A \& Id^\alpha)$. Clearly $C^\alpha_R(A)$ will be the smallest regular $\mathcal{E}^\alpha$-theory containing $A$; this is also the smallest regular $\mathcal{L}^\alpha$-theory containing $A$ (Sec. 2.11). That is, $C^\alpha_R(A) = C^\alpha(A \& Id^\alpha) = C^\alpha(A \& Id^\alpha) = C^\alpha((A \& Id^\alpha) \circ Id^\alpha)$.

This relationship is easy to demonstrate directly for the Detachment systems we are interested in here, by showing that $A \& Id^\alpha \leftrightarrow (A \& Id^\alpha) \circ Id^\alpha$ is provable. Right to left follows by Detachment and $M \&$. For the other direction, note that $A \& B \Rightarrow (A \& B) \circ B$ will be provable in general: $A \& B \Rightarrow A \& B \& B$ (by simple moves) $\Rightarrow (A \& B) \circ B$ (using the derived ordering principle $A \& B \Rightarrow A \circ B$).

**Theorem 70** Let $A$ and $B$ be sentences in the full positive fragment of $\mathcal{L}$ such that $A$ has no positive occurrences of $\neg$, and $B$ no negative occurrences. Let $A'$ and $B'$ result from $A$ and $B$ by replacing all occurrences of $\neg$ by $\circ$. Let $Id_1$ be the conjunction of all instances of Identity in the full positive vocabulary of $\mathcal{L}$, and $Id_2$ be the conjunction of all instances in the positive vocabulary without $\neg$. Then $A \otimes Id_1 \Rightarrow B$ is provable in $\mathcal{E}^{\neg \rightarrow DT}_{D_2E}$ iff $A' \otimes Id_2 \Rightarrow B'$ is provable in $\mathcal{E}^{\neg \rightarrow DT}_{DEF}$, where $\otimes$ is (consistently) either $\circ$ or $\&$.

**Proof** By Th. 7, if $A \otimes Id_1 \Rightarrow B$ is provable in $\mathcal{E}^{\neg \rightarrow DT}_{D_2E}$, $A' \otimes Id_2 \Rightarrow B'$ is provable in $\mathcal{E}^{\neg \rightarrow DT}_{D_2E}$; it is easy to see that $Id_2 \leftrightarrow Id_2'$ is also provable in this system and the provability of $A' \otimes Id_2 \Rightarrow B'$ follows easily. For the other direction, if $A' \otimes Id_2 \Rightarrow B'$ is provable in the smaller system then Th. 9 tells us that $A \otimes Id_2 \Rightarrow B$ is provable in the larger, and $Id_1 \Rightarrow Id_2$ is clearly also provable. The provability of $A \otimes Id_1 \Rightarrow B$ follows straightforwardly.

**Theorem 71** Let $A$ and $B$ be sentences in the full language $\mathcal{L}$. Let $Id_1$ be the conjunction of all instances of Identity in this vocabulary and let $Id_2$ be the corre-
sponding conjunction over the full positive vocabulary (extended with \( \neg \) occurring only in atomic contexts, in an inactive way, as in Sec. 3.5). Then \( A \otimes D_1 \Rightarrow B \) is provable in \( \mathcal{E}^{D\text{TEF}} \) iff \( A^D \otimes D_2 \Rightarrow B^D \) is provable in \( \mathcal{E}^{D\text{TEF}} \), where \( \otimes \) is (consistently) either \( \circ \) or \( \& \).

Proof The argument here follows the same lines as that for the previous theorem, except that we use Th. 44 as appropriate.

As the final result in this series, and following in the same fashion, we have:

**Theorem 72** Let \( \mathcal{E}^\circ \) be a conservative extension of \( \mathcal{E}^\beta \). Let \( D_1 \) and \( D_2 \) be the conjunction of all instances of Identity in the two vocabularies respectively. Let \( A \) and \( B \) be sentences in the vocabulary of the smaller system \( \mathcal{E}^\beta \). Then \( A \otimes D_1 \Rightarrow B \) is provable in \( \mathcal{E}^\circ \) iff \( A^D \otimes D_2 \Rightarrow B^D \) is provable in \( \mathcal{E}^\beta \), where \( \otimes \) is (consistently) either \( \circ \) or \( \& \).

These three results produce the following:

**Theorem 73** Let \( A \) and \( B \) be sentences in the full vocabulary of \( L \) such that \( A^D \) and \( B^D \) are restricted in the following ways:

- They contain no occurrences of \( + \).
- \( A^D \) contains no positive occurrences of \( \neg \), and \( B^D \) no negative occurrences.

Let \( A^* \) and \( B^* \) result from \( A^D \) and \( B^D \) by replacing all occurrences of \( \neg \) by \( \circ \). Then

- \( B \in C_L^{D\text{TEF}}(A) \) iff \( B^* \in C_L^{O\text{DTE}}(A^*) \).
- \( B \in C_R^{D\text{TEF}}(A) \) iff \( B^* \in C_R^{O\text{DTE}}(A^*) \).

This result determines the fragment of \( \mathcal{Q}L^{D\text{TEF}} \) which we can represent using (regular and non-regular) \( RQ_4 \)-theories. This is not the fragment computed by our system, which is even more restricted, but does provide a framework within which to view it. We leave further discussion of the issues until Sec. 4.3.5.
4.3.2 Gentzen Formulations

We mentioned at the beginning of the last section a hiatus in our development. It is this: the logical framework for our system will not be $\mathcal{RQ}_+$, but instead its closely related cousin $\mathcal{LRQ}_+$. $\mathcal{LRQ}_+$ results from our Hilbert-style axiomatization of $\mathcal{RQ}_+$ in Sec. 2.10.7 by dropping the Distribution axiom A4 and the Confinement axioms Q5 and Q6; alternatively, in our entailment system development, by dropping the $D\&\lor$, $D\&\exists$ and $A\lor\forall$. This is sufficient to make propositional $\mathcal{LR}$ decidable, in contrast to $\mathcal{R}$ (which is its main interest for relevant logicians; see [TMM88] for an extensive discussion of $\mathcal{LR}$). We strongly believe that the two systems are identical on the fragment actually computed by our model; annoyingly, a proof still eludes us. The reason for basing our work on $\mathcal{LRQ}_+$ rather than $\mathcal{RQ}_+$ is that it has a very much simpler Gentzen formulation (see below, and compare with Dunn’s Gentzenization of propositional $\mathcal{R}_+$ in [AB75]). On the other hand, the reasons for not couching our previous logical development in terms of $\mathcal{LRQ}$ rather than $\mathcal{RQ}$ are twofold: first, we do not think that dropping Distribution is particularly well motivated, and second, we would be extremely surprised (and disappointed) to find that its absence made any difference to the fragment actually computed. We should stress that doing without Distribution will not affect in any material way the major results developed up to this point; all connections between $\mathcal{RQ}$ and our entailment systems hold equally for $\mathcal{LRQ}$ and the systems which result if we drop the principles mentioned. We apologize for any inconvenience caused by these factors which should be within our control.

For the rest of this Chapter we will only be concerned with first-order systems, and for convenience we will use simple ‘$\mathcal{LR}_+$’ instead of ‘$\mathcal{LRQ}_+$’ to denote the quantified system.

For the task of presenting a logical basis for the computation it will be convenient to work with a Gentzen formulation of $\mathcal{LR}_+$; it is probably the best proof-theoretic vantage point from which to view the logic and describe the peculiarities of our fragment, and it allows a formulation very close to the computation tree model.
The objects of a Gentzen system are sequents of the form $\alpha \vdash \beta$ where $\alpha$ and $\beta$ are some bunchings of formulas, interpreted as '$\beta$ is a consequence of $\alpha$'. For our systems, $\beta$ will always be a single formula, and $\alpha$ will be a multiset of formulas. The interpretation of $\alpha \vdash A$ in our systems is then: 'A follows from $\alpha$ in a way which uses every element of $\alpha$'. The requirement that all elements of $\alpha$ contribute to $A$ is the distinguishing mark of relevance.

Before proceeding we will give some definitions and terminology for multisets. A multiset is like a set except that there may be repetitions of elements. Conversely, a multiset is like a sequence except that the order of elements is unimportant. We will denote multisets with elements enclosed in square brackets: e.g. $[A, B, B, C]$. $[A, B, B] \neq [A, B]$ (since the first has two occurrences of B and the second only one), but $[A, B] = [B, A]$ since order is unimportant. We may also denote multisets with small Greek letters $\alpha, \beta, \gamma...$ where a listing of the elements is not important. Multiset union will be denoted by '+' $[A, B], [B, C] = [A, B, B, C]$. In general, if $A$ occurs $m$ times in $\alpha$ and $n$ times in $\beta$ then it occurs $m + n$ times in $\alpha, \beta$. 'A' may be used to as shorthand for the multiset $[A]$. The generator of a multiset is the set of elements occurring in it: so the generator of $[A, B, C, B]$ is $\{A, B, C\}$. If the generator of $\alpha$ is a subset of the generator of $\beta$, and for every $A$ occurring $m$ times in $\beta$, $A$ occurs $n \leq m$ times in $\alpha$, then $\alpha$ is a weak submultiset of $\beta$. In the case where the two have the same generator, we say that $\alpha$ is a strong submultiset of $\beta$.

A sequent of the form $\vdash A$ has the empty multiset on the left of the '$\vdash$'.

The System $GLR_+$

Axioms

(Id) $p \vdash p$

where $p$ is an atom.

($\vdash t$) $\vdash t$
**CHAPTER 4. RELEVANT LOGIC PROGRAMMING**

Rules

\[
\begin{align*}
(\rightarrow \vdash) & \quad \alpha \vdash A \quad \beta, B \vdash C \\
& \quad \alpha, \beta, A \rightarrow B \vdash C \\
(\circ \vdash) & \quad \alpha, A, B \vdash C \\
& \quad \alpha, A \circ B \vdash C \\
(& \vdash) & \quad \alpha, A \vdash C \\
& \quad \alpha, A \& B \vdash C \\
(\& \vdash) & \quad \alpha, A \vdash C \\
& \quad \alpha, B \& A \vdash C \\
(\lor \vdash) & \quad \alpha, A \lor C \quad \alpha, B \lor C \\
& \quad \alpha, A \lor B \vdash C \\
(V \vdash) & \quad \alpha, A \{s/x\} \vdash B \\
& \quad \alpha, (\forall x) A \vdash B \\
(\exists \vdash) & \quad \alpha, A \{s/x\} \vdash B \\
& \quad \alpha, (\exists x) A \vdash B \\
(W \vdash) & \quad \alpha, A, A \vdash B \\
& \quad \alpha, A \vdash B \\
(\rightarrow \rightarrow) & \quad \alpha, A \vdash B \\
& \quad \alpha \vdash A \rightarrow B \\
(\rightarrow \circ) & \quad \alpha \vdash A \quad \beta \vdash B \\
& \quad \alpha, \beta \vdash A \circ B \\
(\rightarrow \& ) & \quad \alpha \vdash A \quad \alpha \vdash B \\
& \quad \alpha \vdash A \& B \\
(V \rightarrow)(a) & \quad \alpha \vdash A \\
& \quad \alpha \vdash A \lor B \\
(\lor \rightarrow)(b) & \quad \alpha \vdash B \lor A \\
(\forall \vdash) & \quad \alpha \vdash A \{s/x\} \\
& \quad \alpha \vdash (\forall x) A \\
(\exists \vdash) & \quad \alpha \vdash A \{s/x\} \\
& \quad \alpha \vdash (\exists x) A \\
(\top \vdash) & \quad \alpha \vdash A \\
& \quad \alpha, \top \vdash A \\
\end{align*}
\]

We have the usual quantifier restrictions: in each quantifier rule, \(s\) is a term free for \(x\) in \(A\); in \((\forall \vdash)s\) \(s\) may not occur free in \(\alpha\); in \((\exists \vdash)s\) \(s\) may not occur free in \(\alpha\) or \(B\).

For each connective in the language (and for the constant \(t\)) we have one rule introducing that element on the left, and one rule introducing it on the right. Everything which appears in the antecedents of a rule also appears in the conclusion. Some other points to note:

- \((\rightarrow \& )\) requires that both premisses have the same antecedent, while \((\rightarrow \circ )\) does not. \((\& \vdash)\) allows the introduction of arbitrary elements; \((\circ \vdash)\) on the other hand only allows the bunching of things which are relevant to the conclusion. Loosely, \(\circ\) on the right is 'easier' than \(&\); on the left this is reversed.

- \((\rightarrow \rightarrow )\) gives us the basis of a method for computing implicational queries; to prove \(A \rightarrow B\) we must prove \(B\) in a way which uses \(A\). This is the Gentzen form of the Ramsey Test of Chapter 1.
• \((\to \vdash)\) is our main target in simplifying the system down to the level realized by the computation. Essentially, we want to replace it by a single premiss rule; more discussion on this point later.

A \(\mathcal{GLR}_+\) proof of a sequent \(\alpha \vdash A\) is a finite tree of sequents, with \(\alpha \vdash A\) at the root and instances of \((Id)\) at the tips. Every node other than a tip node follows from its children by an application of some \(\mathcal{GLR}_+\) rule.

**Theorem 74 (Admissibility of Cut)** Whenever

\[
\alpha \vdash A
\]

and

\[
\beta, A \vdash B
\]

are both provable in \(\mathcal{GLR}_+\) then so is

\[
\alpha, \beta \vdash B.
\]

See [Mey66, §II.A] for the modifications needed to the argument of [Cur77, §V.D] for the proof of this theorem.

Given Cut we can also show the admissibility of the following principles:

\[
\begin{align*}
(\text{Transitivity}) & \frac{\alpha \vdash A \quad A \vdash B}{\alpha \vdash B} \\
(\text{Detachment}) & \frac{\alpha \vdash A \quad \beta \vdash A \to B}{\alpha, \beta \vdash B}
\end{align*}
\]

**Theorem 75** \(\vdash A\) is provable in \(\mathcal{GLR}_+\) iff \(A\) is a theorem of \(\mathcal{LR}_+\)

**Proof**

• Right to left. It is straightforward to show that for every axiom \(A\) of \(\mathcal{LR}_+\) (cf the Hilbert formulations of Chapter 2), \(\vdash A\) is provable in \(\mathcal{LR}_+\). If
\( \vdash A \) and \( \vdash B \) are both provable then so is \( \vdash A \& B \), by the \((\vdash \&)\) rule. If \( \vdash A \) and \( \vdash A \rightarrow B \) are both provable, then so is \( \vdash B \), by \textit{Cut}. Finally, if \( \vdash A \) is provable then so is \( \vdash (\forall x)A \), by \((\vdash \forall)\). So the class of \( A \) for which \( \vdash A \) includes all of the axioms of \( LR_+ \) and is closed under Adjunction, Detachment and Universalization and hence includes all theorems of \( LR_+ \).

- Left to right. Define a translation mapping sequents into formulas as follows:

\[
\begin{align*}
- T(\vdash A) &= A. \\
- T([A_1 \ldots A_m] \vdash B) &= A_1 \circ \ldots \circ A_m \rightarrow B, \text{ where } m \geq 1.
\end{align*}
\]

It is then a straightforward induction on the \( GLR_+ \) proof to show that \( T(\alpha \vdash A) \) is a theorem of \( LR_+ \) whenever \( \alpha \vdash A \) is provable, and the result follows immediately.

As a corollary:

**Theorem 76** Let \( C_L \) be the logic-based consequence operation of Sec. 4.3.1 tailored for the entailment system which results when we drop \( D \& \forall \), \( D \& \exists \) and \( A \forall \forall \) from \( \mathcal{E}_{OPT}\). Then \( \alpha \vdash B \) is provable in \( GLR_+ \) iff \( B \in C_L(A) \), where we read \( \gamma \) (in the Gentzen formulation) as equivalent to the conjunction of all instances of Identity in the vocabulary of the entailment system, and recalling that the occurrence of \( A \) in \( \langle A \vdash B \rangle \) stands for the multiset consisting of a single instance of \( A \).

### 4.3.3 Towards a Computationally Viable Fragment

As we intimated above, the full system \( GLR_+ \) is too expensive to form the basis of a logic programming language, so we will have to simplify it, losing some expressive power in the process. The goal is to find a fragment which will support our computational model while retaining as much of the power of the full system as possible, in particular allowing full nesting of implications.
We take it that the essential characteristics of ‘linear’ proof systems like SLD and SLC are as follows:

**P1** At any point in the computation we need only work on reducing the current ‘goal’: in Gentzen terms this means essentially that for any connective $\oplus$ if a sequent $\alpha \vdash A \oplus B$ is both allowed by the fragment and provable then there must be a proof of it in which the last step is an application of ($\vdash \oplus$) (or ($W \vdash$), but we can ignore this complication for now). To use the standard terminology, whenever $\oplus$ can occur as the major connective of a goal, the rule ($\vdash \oplus$) must be invertible (again ignoring the question of ($W \vdash$)). In full $GLR_+$, however, the only right hand rules with this property are ($\vdash \forall$), ($\vdash \&$) and ($\vdash \rightarrow$).

**P2** In trying to prove an atom $p$ from $\alpha$ it is enough to find rules $A \rightarrow p$ in $\alpha$, for which $A$ is provable. Although we require the expressive power of the system to include full nesting of implications, we only allow assertions of the form $A \rightarrow p$ where $p$ is an atom. This means that our fragment must allow the conversion of any $A \rightarrow B$ into an equivalent form with an atomic consequent.

These requirements provide almost all the constraints we need to define the fragment.

P2 suggests that we find a fragment for which the complicated ($\rightarrow \vdash$) rule can be replaced by:

$$
\frac{\alpha \vdash A}{\alpha, A \rightarrow p \vdash p}
$$

On the other hand, since we require that the expressive power of the system include full nesting of implications while at the same time allowing only atomic consequents, we will have to find an operation by which $A \rightarrow B \rightarrow C$, for example, can be converted to a logically equivalent form $D \rightarrow p$ for some $D, p$ where $p$ is an atom. For non-relevant systems like NPROLOG we can use the following equivalences.
• $A \supset B \supset C \iff A \& B \supset C$

• $A \supset B \& C \iff (A \supset B) \& (A \supset C)$

where $\supset$ is a nonrelevant implication. But $A \rightarrow .B \rightarrow C$ is not logically equivalent to $A \& B \rightarrow C$ in relevant systems; if it were the archetypical irrelevancy $A \rightarrow .B \rightarrow A$ would follow instantly since $A \& B \rightarrow A$ is a theorem. Specifically, $A \& B \rightarrow C$ follows from $A \rightarrow .B \rightarrow C$, but not vice-versa, since $B$ might be totally irrelevant to $C$.

What we do have in $LR_+$ is the equivalence of $A \rightarrow .B \rightarrow C$ and $A \circ B \rightarrow C$; we can remove right nested implications by exporting out the nested antecedent in a fusion kind of way, rather than with $\&$. On the other hand the second equivalence does hold relevantly (but only for $\&$, not for $\circ$).

For the moment, let us loosely define the **query parts** of a formula (either an assertion or a query) as those subformulas which might occur as a query during the course of an SLC proof. For example, in an assertion $A \land B \rightarrow C$, $A \land B$ is a query part, since this rule might be used in a computation to rewrite a query $C$ to $A \land B$. Dually, we define the assertion parts to be subformulas which might occur as an assertion (ie, as one of the elements of a database segment) so for example in a query $A \rightarrow B$, $A$ is an assertion part, since the method for reducing implicational goals involves the formation of a new segment from the antecedent of the implication. The expression of an assertion $A \rightarrow .B \rightarrow C$ as $A \circ B \rightarrow C$ means that $\land$ must have at least some of the effects of $\circ$, since $\circ$ will have to be allowed to occur in query parts. Let us strengthen this observation, with the following provisional stipulation (subject to later analysis):

**Query Stipulation** $\land$ in query parts is $\circ$.

How does this fit with our invertibility requirement P1? For a start, the rule $(\vdash \circ)$ is not invertible in full $LR_+$; for example there is no proof of $A \circ B \vdash A \circ B$ in which the last step is an application of $(\vdash \circ)$ (or $(W \vdash)$). As it turns out, however, the required property holds if we restrict ourselves to sequents which also obey this second stipulation:

**Assertion Stipulation** $\land$ in assertion parts is $\&$
In other words, we take it that within segments elements are bunched with & rather than o. To put the case in a more concrete form, consider the 'base-level' segment we get when we assert A to the system, and then B. We would expect the query A to succeed from this database, which means that we think of it as A&B rather than as AoB, since AoB ⊨ A is not provable (because the B contributes nothing to A). So the assertion stipulation is natural for the 'base-level' segment, and we extend it to all the segments generated in the course of the computation.

In this discussion we have so far ignored the question of the invertibility of (⊢ ∨) and (⊢ ∃) (which are not invertible in the full system GLR⁺); to achieve this we will need to carry over the normal PROLOG restriction disallowing disjunctions in assertion parts. This is probably the most serious restriction to the expressive power of the system, but unavoidable if we wish to retain the simple notion of an 'answer substitution' and the PROLOG rule for computing disjunctive queries.

The conjunctive stipulations may be judged according to the extent to which the fragment they determine meets the following criteria:

- Expressive power.
- Naturalness.
- Fulfillment of the computational objectives.

We show in later sections that they pass the third of these tests; for the moment we will concentrate on the first two.

Expressive Power and Naturalness of the Conjunctive Restrictions

We first consider the issues for queries. A query A ∧ B is taken implicitly as AoB, which might be a little disturbing until one realizes that for a single element multiset α (corresponding to a single segment database) obeying the Assertion Stipulation, α ⊨ AoB iff α ⊨ A&B, and so ∧ behaves as expected for normal PROLOG situations. On the other hand, this would not be the case without the Assertion Stipulation; for instance AoB ⊨ A&B does not hold. A query A→B∧C
CHAPTER 4. RELEVANT LOGIC PROGRAMMING

will be taken as $A \rightarrow BoC$; at the base-level, with the Assertion Stipulation, we can express the analogous query with $\&$ in the consequent using these facts: 

\[ \alpha \vdash A \rightarrow B \& C \text{ iff } \alpha \vdash (A \rightarrow B) \& (A \rightarrow C) \text{ iff } \alpha \vdash (A \rightarrow B) \circ (A \rightarrow C) \text{ iff } \alpha \vdash (A \rightarrow B) \land (A \rightarrow C). \]

At higher levels we can iterate this process of decomposing consequents to the same effect. On the antecedent side, a query $A \& B \rightarrow C$ translates as $A \& B \rightarrow C$, i.e., with a sense that does not require both $A$ and $B$ to have some connection to $C$. If we want the stronger query $A \circ B \rightarrow C$ we can express it in the logically equivalent form $A \rightarrow B \rightarrow C$. As far as queries are concerned, then, the reduction in expressive power seems quite small; essentially all we are prevented from asking is things like $A \rightarrow Bo(C \& D)$ and $A \& (BoC) \rightarrow D$ (i.e., with $\&$ occurring in the ‘scope’ of $\circ$ in a query part, and conversely for an assertion part); we find it hard to get upset about this. The question of whether the translation of $\&$ in each circumstance is the most natural reading of ‘and’ is quite subjective and probably not well-posed, since natural usage of ‘and’ does not seem totally coherent: if we ask the question ‘Does $C$ follow from $A$ and $B$?’ do we generally mean this in a sense for which both $A$ and $B$ must contribute to $C$, or do we merely mean to ask whether there is ‘enough information’ in the set $\{A, B\}$ to deliver $C$? At least the system allows us to make these fine distinctions, which are obliterated by the rough brush of irrelevance. Similarly, we can make the distinction between two senses of ‘If we build a factory in zonel, will zone2 be polluted and dangerous?’ The first of these is one which requires building a factory to lead to both situations; the second, which is taken by our translation of $\&$, is weaker and requires only that the factory contribute to one of the effects, pollution for example, with the danger perhaps caused by other things. Both senses seem reasonably common, both can be expressed in this language and again it is at least some increase in expressive power to be able to make the distinction.

On the assertion side, the restrictions in expressive power are more limiting but at the same time perhaps less important, since the natural usages seem more firmly fixed. For a start, we cannot express an assertion $A \circ B$ but this is hardly a worry; neither do we regret the lack of an equivalent to the assertion $A \rightarrow BoC$. $A \rightarrow B \& C$ may of course be expressed by $(A \rightarrow B) \land (A \rightarrow C)$, and $A \rightarrow .B \rightarrow C$ by...
$A \land B \rightarrow C$. Most serious is the inability of the language to express an assertion $A \& B \rightarrow C$ rather than $A \circ B \rightarrow C$, but it seems to us that generally when we assert that ‘$A$-and-$B$’ implies $C$ we mean that both $A$ and $B$ have something to do with $C$. If not, why include the irrelevant element? See Sec. 4.4 for some further discussion.

4.3.4 Formalizing the Fragment

In this section we give a formal account of the fragment of $LR_+$ computed by SLC. First we introduce syntactic restrictions on the occurrence of various connectives, yielding a class of sequents which we call the $L/R$ (‘Left/Right’) sequents. The $L/R$ restrictions satisfy our requirement P1; if an $L/R$ sequent $\alpha \vdash A \oplus B$ is provable then there is a proof in which the last step is an application of either ($\vdash \oplus$) or ($W \vdash$). Then we introduce a normal forming operation on $L/R$ sequents, converting them into $A/Q$ (‘Assertion/Query’) form; this mainly involves the translation of assertions into equivalent forms with atomic consequents in the manner discussed above, satisfying the second requirement P2. Finally we introduce a new Gentzen formulation - $\mathcal{GLR}_{AQ}$ - with our simplified form of the $(\rightarrow \vdash)$ rule, and show that an $L/R$ sequent is provable in $\mathcal{GLR}_+$ iff the corresponding $A/Q$ sequent is provable in $\mathcal{GLR}_{AQ}$. In the next section we demonstrate the equivalence of $\mathcal{GLR}_{AQ}$ and the computation tree model.

L/R Form

The major restrictions on expressive power in the fragment of $LR_+$ to be considered fall into three classes:

- Quantifier restrictions
- Disjunctive restrictions
- Conjunctive restrictions
The quantifier and disjunctive restrictions are natural extensions to the normal PROLOG restrictions: no disjunctions or existential quantifiers in assertions, and no universal quantifiers in queries. The conjunctive restrictions follow from the considerations of the previous section. Here we have generalized notions of assertions and queries: formulas put as queries to the system may have subformulas which will act like assertions in the course of the computation, and dually for formulas asserted to the system.

We will need briefly the following notion of surface parts, essentially those parts of a formula which are not buried in the antecedents or consequents of implications:

**Definition 13 (Surface Parts)**

- Any occurrence of a formula is a surface part of itself.
- If \( B \oplus C \) is a surface part of \( A \), where \( \oplus \) is either \( \& \), \( \vee \) or \( \circ \), then so are the occurrences of \( B \) and \( C \).

So the surface parts of \( A \&(B \circ(C \rightarrow D)) \) are (the occurrences of) \( A \&(B \circ(C \rightarrow D)) \), \( A \), \( B \circ(C \rightarrow D) \), \( B \) and \( C \rightarrow D \), but not \( C \) or \( D \).

**Definition 14 (Left and Right Wffs)** Define two classes of formulas, left wffs and right wffs, characterized by dual restrictions on occurrences of \( \circ \), \( \& \), \( \vee \), \( \forall \), \( \exists \) and \( t \). In the table, ‘+’ means that the language element may only occur positively, ‘-’ that it may only occur negatively:

<table>
<thead>
<tr>
<th></th>
<th>Left Wffs</th>
<th>Right Wffs</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \circ )</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>( &amp; )</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>( \vee )</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>( \forall )</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>( \exists )</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>( t ) restriction</td>
<td>-</td>
<td>+</td>
</tr>
</tbody>
</table>

The \( t \) restriction is a stipulation that subformulas like \( A \rightarrow B \), where \( t \) is a surface part of \( B \), may only occur negatively in left wffs and positively in right wffs.
Left wffs provide the basic material for the class of formulas which may be asserted to our extended PROLOG; dually, right wffs will determine the kind of queries which may be put to the system. In the next section we define normal forming operations on left and right wffs, converting them into the forms appropriate for computational model.

The reason for the awkward \( t \) restriction is to disallow assertions like \( A \rightarrow t \) (note that this is cannot be a left wff under the above definition), because while according to the logic, \( A \rightarrow p \rightarrow p \) (for example) should follow from a database containing \( A \rightarrow t \) (remembering that \( t \) is equivalent to the conjunction of all \( B \rightarrow B \)), supporting this in the computational model would be difficult.

Example of a left wff: \( (\exists x)(f(x) \circ (\exists y)(g(x,y) \lor h(y))) \rightarrow (\forall z)(g(z,a) \& h(z)) \)

Example of a right wff: \( (\forall z)(g(z,a) \& h(z)) \rightarrow (\exists x)(f(x) \circ (\exists y)(g(x,y) \lor h(y))) \)

These examples are intended to illustrate the following fact: if \( A \rightarrow B \) is a left (right) wff, so is \( B \), and \( A \) is a right (left) wff. In other word, positive parts of left (right) wffs are left (right) wffs, and negative parts are right (left) wffs.

For completeness, we give the explicit syntax:

- An atom \( p \) (including \( t \)) is both a left and a right wff.
- If \( A \) and \( B \) are left wffs then so is \( A \& B \).
- If \( A \) and \( B \) are right wffs then so are \( A \circ B \) and \( A \lor B \).
- If \( A \) is a right wff and \( B \) is a left wff s.t. \( t \) does not occur as a surface part of \( B \) then \( A \rightarrow B \) is a left wff.
- If \( A \) is a left wff and \( B \) is a right wff then \( A \rightarrow B \) is a right wff.
- If \( A \) is a left wff then so is \( (\forall x)A \).
- If \( A \) is a right wff then so is \( (\exists x)A \).

**Definition 15 (L/R Sequent and Proofs)**

- A sequent \( \alpha \vdash A \) is an L/R sequent iff every element of \( \alpha \) is a left wff and \( A \) is a right wff.
• A $\mathcal{GLR}_+$ proof is an L/R proof iff every sequent in it is an L/R sequent.

Theorem 77  
• Every $\mathcal{GLR}_+$ proof of an L/R sequent is an L/R proof.

• No L/R proof contains any applications of the rules ($\vdash \&$), ($\circ \vdash$), ($\forall \vdash$), ($\lnot \forall$) and ($\exists \vdash$).

The proof of this theorem rests on the facts that none of the $\mathcal{GLR}_+$ rules yield L/R sequents from premisses which are not L/R sequents, and that the conclusion of any application of the rules listed is never an L/R sequent.

The following theorem establishes that the L/R restrictions satisfy our requirement P1. In effect, if there is a proof of an L/R sequent then there is a proof which can be divided horizontally into left and right parts - in the top part only left rules are used, and in the bottom only right rules and ($W \vdash$). (There are more complicated Gentzen formulations of $\mathcal{LR}_+$ which build the effects of ($W \vdash$) into the other rules - in such a system the division between left and right would be complete).

Theorem 78 (Quasi-Inversion) Let $A$ be a non-atomic formula and let the major connective of $A$ be $\oplus$. Let $\alpha \vdash A$ be an L/R sequent. If there is a $\mathcal{GLR}_+$ proof of $\alpha \vdash A$ then there is a $\beta$ s.t. $\alpha$ is a strong sub-multiset of $\beta$, and there is a $\mathcal{GLR}_+$ proof of $\beta \vdash A$ in which the last step is an application of ($\vdash \oplus$).

The proof of this theorem may be extracted from [Cur77, §5.D].

As a consequence, whenever $\alpha \vdash A \lor B$ is provable, so is either $\alpha \vdash A$ or $\alpha \vdash B$; and generalizing, if $\alpha \vdash (\exists x)A$ is provable then so is $\alpha \vdash A\{s/x\}$ for some s. For the theorem tells us that when $\alpha \vdash A$ is provable, and the major connective of $A$ is $\oplus$, for which the ($\vdash \oplus$) rule has a single premiss $\alpha \vdash A'$, then $\beta \vdash A'$ is provable for some $\beta$ s.t. $\alpha$ is a strong sub-multiset of $\beta$. But then $\alpha \vdash A'$ follows by steps of ($W \vdash$). This means that we will be able to retain the PROLOG rule for computing disjunctive queries, and the simple notion of an ‘answer substitution’.

Note that this is not true for $\mathcal{GLR}_+$ proofs in general; for example, $f(a) \lor f(b) \vdash$
(\exists x)f(x) is provable but there is no s for which \( f(a) \lor f(b) \vdash f(s) \) is provable. As regard conjunctive goals, the theorem means that a proof of \( A \land B \) (i.e., \( A \lor B \), with the query stipulation) can be reduced to independent proofs of \( A \) and \( B \) (with an incrementally constructed answer substitution).

Q-Normal Forms

Q-normal left and right wffs are normalized forms of left and right wffs in which quantifiers have been brought to the front of the formula. For completeness, we give the explicit syntax:

- A Q-normal left wff has the form \( \forall A \) where \( A \) is an unquantified left wff and \( \forall \) is a string (perhaps null) of distinct universal quantifiers.
- A Q-normal right wff has the form \( \exists A \) where \( A \) is an unquantified right wff and \( \exists \) is a string (perhaps null) of distinct existential quantifiers.
- An unquantified left wff is either an atom or a formula of the form \( A \rightarrow B \) where \( A \) is an unquantified right wff and \( B \) is an unquantified left wff, or a formula of the form \( A \land B \) where \( A \) and \( B \) are unquantified left wffs.
- An unquantified right wff is either an atom, or a formula of the form \( A \lor B \) or \( A \lor B \) where \( A \) and \( B \) are unquantified right wffs, or a formula of the form \( A \rightarrow B \) where \( A \) is a Q-normal left wff, and \( B \) is an unquantified right wff.

Clearly, every Q-normal left (right) wff is a left (right) wff. Note that internal quantifiers are allowed in the Q-normal forms, but only in contexts \( (\forall x)B \rightarrow C \) occurring positively in Q-normal right wffs and negatively in the left forms.

Definition 16 (Q-Normalization) In what follows we will let \( \forall \) and \( \exists \) (possibly with subscripts) stand for arbitrary strings of universal and existential quantifiers. Define an operation \( (\cdot)^L \) mapping left wffs into Q-normal left wffs, and the corresponding operation \( (\cdot)^R \) for right wffs as follows:
1 As a first step in both cases, rename all variables in the formula so that no variable occurs both bound and free in any subformula, and each variable is bound by at most one quantifier. For example, $(\exists x)A \circ (\exists x)B$ becomes $(\exists x)A \circ (\exists y)B \{y/x\}$ for some $y$ not occurring in $A$ or $B$.

2 $p^L = p^R = p$ for all atoms $p$.

3 $(A \& B)^L = \forall_1 \forall_2 (A' \& B')$ where $A^L = \forall_1 A'$ and $B^L = \forall_2 B'$.

4 $(A \oplus B)^R = \exists_1 \exists_2 (A' \oplus B')$ where $A^R = \exists_1 A'$ and $B^R = \exists_2 B'$, and $\oplus$ is either $\lor$ or $\rightarrow$.

5 $((\forall x)A)^L = (\forall x)A^L$.

6 $(\exists x)A)^R = (\exists x)A^R$.

7 $(A \rightarrow B)^L = \forall_1 \forall_2 (A' \rightarrow B')$ where $B^L = \forall_2 B'$ and $A^R = \exists_1 A'$, and $\forall_1$ binds the same variables as $\exists_1$.

8 $(A \rightarrow B)^R = \exists (A^L \rightarrow B')$ where $B^R = \exists B'$.

9 Where $\alpha$ is a multiset of left wffs $[A_1 \ldots A_m]$ we let $\alpha^L = [A_1^L \ldots A_m^L]$.

It is easy to see that the translation maps left wffs into Q-normal left wffs and right wffs into Q-normal right wffs.

All of the steps preserve $LR$ equivalence, except for step 8, since it is not generally the case that $A \rightarrow (\exists x)B$ and $(\exists x)(A \rightarrow B)$ are equivalent in relevant systems. In other systems (but not intuitionism) it exists as the generalization of the relevantly undesirable equivalence of $A \rightarrow B \lor C$ and $(A \rightarrow B) \lor (A \rightarrow C)$, which leads to theorems like $(A \lor B \rightarrow A) \lor (A \lor B \rightarrow B)$ where we have no reason to believe either disjunct. However, with the restrictions we operate under, step 8 causes no problems; for our purposes it is enough that the following holds:

**Theorem 79** Let $\alpha$ be a multiset of left wffs and let $A \rightarrow B$ be a right wff, such that $x$ does not occur in $A$ or $\alpha$. Then $\alpha \vdash A \rightarrow (\exists x)B$ is provable in $GLR_+$ iff $\alpha \vdash (\exists x)(A \rightarrow B)$ is.
Proof Right to left is relevantly acceptable for unrestricted wffs. So assume that \( \alpha \vdash A \rightarrow (\exists x)B \) is provable in \( {\mathcal G LR}_+ \). Using Cut we can show that there is a proof of \( \alpha, A \vdash (\exists x)B \), which is an L/R sequent, and so by Th. 78 there is a proof of \( \alpha, A \vdash B\{s/x\} \) for some \( s \). \( \alpha \vdash A \rightarrow B\{s/x\} \) follows by \( (\vdash \rightarrow) \), but since \( x \) does not occur in \( A \), this to say that \( \alpha \vdash (A \rightarrow B)\{s/x\} \) is provable and the result follows by \( (\vdash \exists) \).

Given this, the following result is straightforward:

**Theorem 80** There is a \( {\mathcal G LR}_+ \) proof of an L/R sequent \( \alpha \vdash A \) iff there is a proof of \( \alpha^L \vdash A^R \).

Readers might wonder whether all internal quantifiers could not be eliminated by translating a right wff of the form \( (\forall x)B \rightarrow C \) to \( (\exists x)(B \rightarrow C) \), for example. As in the translation of Step 8 above, this is not generally valid for relevant logics (in other logics it holds as the generalization of the equivalence of \( A \& B \rightarrow C \) and \( (A \rightarrow C) \lor (B \rightarrow C) \)). Unlike Step 8, however, the L/R restrictions do not produce a system in which the move is admissible; for example, \( f(a) \circ f(b) \rightarrow g \vdash (\forall x)f(x) \rightarrow g \) is provable but there is no term \( s \) for which \( f(a) \circ f(b) \rightarrow g \vdash f(s) \rightarrow g \) is.

**A/Q Form**

We have defined the expressive power of the system in terms of L/R sequents. In this section we introduce a normal forming operation on L/R sequents, converting them into A/Q ('Assertion/Query') form. An A/Q sequent \( \alpha \vdash A \) has \( \alpha \) as a multiset of assertions and \( A \) as a query. Assertions and queries in this logical discussion are essentially as they were in our computational development; the difference is that we apply the assertion and query stipulations of our motivational discussion to translate \( \wedge \). In queries, \( \wedge \) becomes \( \circ \) in positive positions and \( \& \) in negative positions; the situation is reversed for assertions. Explicitly, the syntax for these modified forms is the same as the syntax for assertions and queries in Sec. 4.2.1 except that items 3 and 4 become:
3 An unquantified assertion has the form $A_1 \& \ldots \& A_m$ where $A_1 \ldots A_m$ are elementary assertions.

4 An unquantified query has the form $A_1 \circ \ldots \circ A_m$ where $A_1 \ldots A_m$ are elementary queries.

Readers can verify for themselves that replacing $\&$ and $\circ$ by $\land$ recovers the previous definitions for queries and assertions; conversely, if in the previous definition we replace $\land$ by $\circ$ in positive occurrences in queries, and by $\&$ in negative occurrences, and dually for assertions, we arrive at the definition given here.

It is straightforward to show that every assertion is a $Q$-normal left wff and every query is a $Q$-normal right wff.

In moving back and forth between the $\land$ and the $\&/\circ$ formulations it will be helpful to have the following notation: for any formula $A$ let $A^\land$ be the result of replacing all occurrences of $\&$ and $\circ$ in $A$ by $\land$.

The next task is to define the normal forming operations.

**Definition 17 (Assertion and Query Translations)** In what follows we will let $\forall$ and $\exists$ (possibly with subscripts) stand for strings of universal and existential quantifiers respectively. Define the assertion translation $(.)^A$, a mapping from left wffs into quantified assertions, and the query translation $(.)^Q$, a mapping from right wffs into quantified queries, as follows:

1. Perform the appropriate $Q$-normalization ($(.)^L$ for left wffs, $(.)^R$ for right wffs).


3. $(A \& B)^A = A^A \& B^A$

4. $(A \oplus B)^Q = A^Q \oplus B^Q$ where $\oplus$ is either $\circ$ or $\lor$.

5. $((\forall x)A)^A = (\forall x)A^A$

6. $((\exists x)A)^Q = (\exists x)A^Q$
7 \((A \rightarrow p)^A = A^Q \rightarrow p\)

8 \((A \rightarrow B \rightarrow C)^A = (A \circ B \rightarrow C)^A\).

9 \((A \rightarrow B \& C)^A = ((A \rightarrow B) \& (A \rightarrow C))^A\).

10 \((A \rightarrow B)^Q = A^A \rightarrow B^Q\)

11 Where \(\alpha\) is a multiset of \(Q\)-normal left wffs \([A_1 \ldots A_m]\) we let \(\alpha^A = [A_1^A \ldots A_m^A]\).

It is straightforward to show that where \(A\) is a left (right) wff, \(A^A (A^Q)\) is defined and is a quantified assertion (query). For example, the steps involved in translating a left wff

\[(\forall x)(\exists y)f(x, y) \rightarrow (\exists z)g(x, z) \rightarrow (\forall z)h(z)\]

into a quantified assertion might be as follows:

- As the first step we form the \(Q\)-normalization.
  - First of all rename variables:
    \[(\forall x)(\exists y)f(x, y) \rightarrow (\exists z)g(x, z) \rightarrow (\forall w)h(w)\]
  - Apply clause 7 of the \(Q\)-normalization to the right nested implication to give:
    \[(\forall x)(\exists y)f(x, y) \rightarrow (\forall z)(\forall w)(g(x, z) \rightarrow h(w))\]
  - Apply clause 7 to the outer implication:
    \[(\forall x)(\forall y)(\forall z)(\forall w)(f(x, y) \rightarrow g(x, z) \rightarrow h(w))\]

This formula is now in \(Q\)-normal form.

- Apply clauses 5 and 8 of the \(A/Q\) translation to give:

\[(\forall x)(\forall y)(\forall z)(\forall w)(g(x, z) \rightarrow h(w))\]

which is a quantified assertion.
The only activity performed in the A/Q translation besides Q-normalization is
the reduction of the consequents of left wffs to atomic form. All of the new
steps preserve $LR$ equivalence, and this together with our results for the Q-
normalization gives the following:

**Theorem 81** There is a $GLR_+$ proof of an L/R sequent $\alpha \vdash A$ iff there is a
proof of $\alpha^A \vdash A^Q$.

**Definition 18 (A/Q Sequents and Proofs)**
- A sequent $\alpha \vdash A$ is an A/Q sequent iff every element of $\alpha$ is a quantified assertion and $A$ is a quantified query.
- A $GLR_+$ proof is an A/Q proof iff every sequent in it is an A/Q sequent.

Clearly every A/Q sequent is an L/R sequent, and every A/Q proof is an L/R proof.

**Theorem 82** Every $GLR_+$ proof of an A/Q sequent is an A/Q proof.

**Proof** No $GLR_+$ rule gives a conclusion which is an A/Q sequent from premisses
which are not, as readers may verify by inspection.

This is an important result (cf the proof of Th. 83 below), and one which demon­
strates the value of a Cut-free Gentzen formulation, in which everything occur­
rning in the premisses of a rule also occurs in the conclusion. Often quite powerful
metatheoretic results can be established by simple inspection of the rules, as here.

We now present another Gentzen system - $GLR_{AQ}$ - which will take us most of
the way towards a the computation tree model, by allowing a simplified version
of $(\rightarrow \vdash)$. In the next section we will compare the two formulations, and prove
them to be equivalent. The major result for $GLR_{AQ}$ is:

**Theorem 83** If $\alpha \vdash A$ is an L/R sequent then there is a $GLR_+$ proof of $\alpha \vdash A$
iff there is a $GLR_{AQ}$ proof of $\alpha^A \vdash A^Q$. 
which shows that the simplified system has the same expressive power as the full system $\mathcal{GLR}_+$ when restricted to the language of our computation.

The system $\mathcal{GLR}_{AQ}$:

**Axioms**

$\text{(Id)} \ p \vdash p$

where $p$ is an atom.

$\vdash t \vdash t$

**Rules**

\[
\frac{\alpha \vdash A}{\alpha, A \rightarrow p \vdash p} \quad \frac{\alpha \vdash A \rightarrow B}{\alpha, A \vdash B} \\
\frac{\alpha, A \vdash C}{\alpha, A \& B \vdash C} \quad \frac{\alpha, A \vdash C}{\alpha, B \& A \vdash C} \\
\frac{\alpha \vdash A \vee B}{\alpha \vdash A \vee B} \quad \frac{\alpha \vdash A \vee B}{\alpha, B \& A \vdash A} \\
\frac{\alpha \vdash A}{\alpha \vdash A} \quad \frac{\alpha \vdash A}{\alpha, A \vdash B} \\
\frac{\alpha \vdash A}{\alpha, A, A \vdash B} \quad \frac{\alpha \vdash A}{\alpha, t \vdash A}
\]

Note that $p$ in $\rightarrow p \vdash$ is an atom (other than $t$). $\mathcal{GLR}_{AQ}$ results from $\mathcal{GLR}_+$ by

- Replacing $\rightarrow t \vdash$ by $\rightarrow p \vdash$.
- Dropping the rules $\circ \vdash$, $\vdash \&$, $\forall \vdash$ and $\exists \vdash$.

In computational terms $\rightarrow p \vdash$ is a great improvement over $\rightarrow t \vdash$. It has a single premiss, and the right-hand side of the conclusion is the same atom as the consequent of the introduced implication. In practical terms, this will allow us to retain the simple backward chaining style of PROLOG.

The proof of Th. 83 uses the following

**Lemma 7 (E.P.Martin)** If there is a $\mathcal{GLR}_+$ proof of an $A/Q$ sequent then there is a proof of the sequent in which all applications of $\rightarrow t \vdash$ have the form
\[
\frac{\alpha \vdash A \quad p \vdash p}{\alpha, A \rightarrow p \vdash p},
\]

where \( p \) is an atom.

Call any proof with this restriction on applications of \((\rightarrow \vdash)\) an \((\rightarrow \vdash)\)-restricted proof.

**Proof of lemma**

Assume there is a \(\mathcal{LR}_+\) proof of an \(A/Q\) sequent \(\alpha \vdash A\). By Th. 82 the proof is an \(A/Q\) proof, and so the result of any application of \((\rightarrow \vdash)\)

\[
\frac{\beta \vdash B \quad \gamma, C \vdash D}{\beta, \gamma, B \rightarrow C \vdash D}
\]

must be an \(A/Q\) sequent. But this means that \(C\) must be an atom (other than \(t\)), because otherwise \(B \rightarrow C\) is not an assertion. It will be enough to demonstrate the following:

If \(\gamma\) has at least one occurrence of an atom \(p\), and if there are \((\rightarrow \vdash)\)-restricted proofs of \(A/Q\) sequents \(\beta \vdash B\) and \(\gamma \vdash C\), then there is an \((\rightarrow \vdash)\)-restricted proof of any \(A/Q\) sequent \(\beta, \delta \vdash C\) where \(\delta\) is like \(\gamma\) except that some or all occurrences of \(p\) are replaced by \(B \rightarrow p\).

(This bundles up an inductive statement with a basis assumption of no applications of \((\rightarrow \vdash)\) in the proofs of \(\beta \vdash B\) and \(\gamma \vdash C\), and the obvious inductive step).

The proof uses induction on the rank of the proof of \(\gamma \vdash C\), relative to the atom \(p\) which has an occurrence in \(\gamma\). This is determined as follows:

- If \(\gamma \vdash C\) is an instance of the axiom \(p \vdash p\) then the rank is 1.
- If \(\gamma \vdash C\) follows by the application of a rule then the rank is one plus the maximum of the ranks of the premisses.
- If the rank is 1 then \(\beta, \delta \vdash C\) is \(\beta, B \rightarrow p \vdash p\) which follows by an application of \((\rightarrow \vdash)\) in the correct form.
• Show that the statement holds for rank \( n \), under the assumption that it holds for all ranks less than \( n \). We have to consider all the possible rules which could have been used to give the last step in the proof of \( \gamma \vdash C \). Note that the proof is an L/R proof and so not all the \( GLR_{+} \) rules apply. To illustrate, consider the case where \( \gamma \vdash C \) follows by \( (\Rightarrow o) \) from \( \gamma_1 \vdash C_1 \) and \( \gamma_2 \vdash C_2 \). By the inductive hypothesis, \( \beta, \delta_1 \vdash C_1 \) and \( \beta, \delta_2 \vdash C_2 \) both have \( (\Rightarrow) \)-restricted proofs (where \( \delta = \delta_1, \delta_2 \)). An application of \( (\Rightarrow o) \) gives \( \beta, \beta, \delta \vdash C \) and the result follows by steps of \( (W \Rightarrow) \). The situation for the other applicable rules is similar or easier.

Proof of Theorem 83

• Assume that there is \( GLR_{+} \) proof of \( \alpha \vdash A \). By the lemma, and by Th. 81 there is an \( (\Rightarrow) \)-restricted proof of \( \alpha^A \vdash A^Q \). Each application of \( (\Rightarrow) \) can be transformed into an application of \( (\Rightarrow p \Rightarrow) \) simply by removing the 'p \Rightarrow p' branch. By Th. 77 the proof does not use any applications of \( (\Rightarrow o) \), \( (\Rightarrow &), (\forall \Rightarrow) \), \( (\forall \Rightarrow) \) or \( (\Rightarrow 3) \) and so the result will be a \( GLR_{AQ} \) proof.

• Assume that there is a \( GLR_{AQ} \) proof of \( \alpha^A \vdash A^Q \). We can transform it into an \( (\Rightarrow) \)-restricted \( GLR_{+} \) proof by adding a \( p \Rightarrow p \) branch to each application of \( (\Rightarrow p \Rightarrow) \), converting it into an application of \( (\Rightarrow) \). The result follows by Th. 81.

4.3.5 Restoring Negation

As we demonstrated in Sec. 4.3.1, \( LR_{+} \) is powerful enough to represent a substantial fragment of our full system with negation \( L^{DTEF} \). We show here how this interacts with the L/R restrictions. We first add the following clause to the definition of positive and negative occurrences (Def. 2):

• If \( A \) occurs positively (negatively) in \( B \) then it occurs negatively (positively) in \( \neg B \).
To avoid confusion here, readers should bear in mind the distinction between true and false occurrences on the one hand, and positive and negative occurrences on the other. For example, in \( A \rightarrow B \), \( A \) occurs \textit{negatively} and \textit{truly} (since \( (A \rightarrow B)^D = A^D \rightarrow B^D \)), whereas in \( \neg(A \rightarrow B) \) it occurs \textit{positively} and \textit{truly} (since \( (\neg(A \rightarrow B))^D = A^D \neg \rightarrow (\neg B)^D \)). On the hand, \( B \) occurs \textit{positively} and \textit{truly} in the first and \textit{negatively} and \textit{falsely} in the second.

We will define extended forms of left and right wffs which will have these properties:

- The NNFs of each extended form will contain no occurrences of \(+\).
- The NNFs of extended left wffs will have only negative occurrences of \( \neg \), and the NNFs of extended right wffs will have only positive occurrences.

We define translations which have the effect of converting extended left and right wffs into NNF, and then replacing occurrences of \( \neg \) by \( o \), in such a way that the results are (simple) left wffs and right wffs. From Ths. 73, 76, 77 and 83 we can conclude that the expressive power of \( \mathcal{GLRAQ} \) is equal to that of \( \mathcal{L}^{DEF} \) in respect of this extended left/right fragment.

**Definition 19 (NLeft and NRight Wffs)** A Left wff with negation (\textit{NLeft wff}) may contain the following kinds of occurrences of language elements (with negation unrestricted):

- True/Positive: \( \& \), \( \rightarrow \), \( \forall \).
- False/Positive: \( + \), \( \& \), \( \rightarrow \), \( \forall \).
- True/Negative: \( o \), \( \forall \), \( \rightarrow \), \( \exists \).
- False/Negative: \( \forall \), \( \exists \).

A Right wff with negation (\textit{NRight wff}) has the following dual conditions:

- True/Positive: \( o \), \( \forall \), \( \rightarrow \), \( \exists \).
CHAPTER 4. RELEVANT LOGIC PROGRAMMING

- **False/Positive**: V, ∃.

- **True/Negative**: &, →, ∀.

- **False/Negative**: +, &, →, ∀.

If we look just at the conditions for true occurrences, we see that we have just the L/R restrictions. For the new elements, no implication is allowed in false, negative occurrences in the extended left wffs; similarly in false, positive occurrences in extended right wffs. Apart from the asymmetry with respect to the conditions for →, the true and false conditions within each positive/negative division are naturally dual to each other.

The explicit syntax may be constructed by adding the following to the clauses for Left and Right wffs in Sec. 4.3.4 (except of course that these now apply to NLeft and NRight wffs):

- If \( A \) is an NLeft (NRight) wff then so is \( \neg \neg A \).
- If \( \neg A \) and \( \neg B \) are both NLeft wffs then so is \( \neg (A \lor B) \).
- If \( \neg A \) and \( \neg B \) are both NRight wffs then so are \( \neg (A \lor B) \) and \( \neg (A \land B) \).
- If \( A \) and \( \neg B \) are both NRight wffs then so is \( \neg (A \rightarrow B) \).
- If \( \neg A \) is an NLeft wff then so is \( \neg (\exists x)A \).
- If \( \neg A \) is an NRight wff then so is \( \neg (\forall x)A \).

Some examples:

- \( \neg (p \rightarrow q) \) is not an NLeft wff, because it has a false, negative occurrence of \( \rightarrow \) (the occurrence of \( \rightarrow \) within \( p \rightarrow q \) is positive, so it becomes negative in \( \neg (p \rightarrow q) \)). The NNF of this formula is \( p \rightarrow \neg q \); when we replace \( \rightarrow \) by \( \circ \) it becomes \( p \circ \neg q \) which is of course not a Left wff.

- \( \neg (p \rightarrow q) \) is an NRight wff; false, negative occurrences of \( \rightarrow \) are allowed here, and \( p \circ \neg q \) is an acceptable form for a right wff.
We can also define QN-normal NLeft and NRight wffs by extending the clauses for Q-normal Left and Right wffs of Sec. 4.3.4 as follows (where we replace ‘Q-normal’ by ‘QN-normal’ and ‘unquantified left/right wff’ by ‘unquantified NLeft/NRight wff’ throughout):

- If $A$ is an unquantified NLeft/NRight wff then so is $\neg \neg A$.
- If $\neg A$ and $\neg B$ are unquantified NLeft wffs then so is $\neg (A \lor B)$.
- If $\neg A$ and $\neg B$ are unquantified NRight wffs then so are $\neg (A \land B)$ and $\neg (A \rightarrow B)$.

The translations $\cdot \overline{\cdot}_{NL}/\cdot \overline{\cdot}_{NR}$ from NLeft/NRight wffs to their QN-normal forms add the following clauses to the Q-normal translation of Sec. 4.3.4:

- $(\neg A)^{NL} = A^{NL}$.
- $(\neg A)^{NR} = A^{NR}$.
- $(\neg (A \lor B))^{NL} = \forall_1 \forall_2 \neg (A' \lor B')$ where $(\neg A)^{NL} = \forall_1 \neg A'$ and $(\neg B)^{NL} = \forall_2 \neg B'$.
- $(\neg (A \oplus B))^{NR} = \exists_1 \exists_2 \neg (A' \oplus B')$ where $\oplus$ is + or $\&$, $(\neg A)^{NR} = \exists_1 \neg A'$ and $(\neg B)^{NR} = \exists_2 \neg B'$.
- $(\neg (A \rightarrow B))^{NR} = \exists_1 \exists_2 (A' \rightarrow B')$ where $A^{NR} = \exists_1 A'$ and $(\neg B)^{NR} = \exists_2 \neg B'$.
- $(\neg (\exists x)A)^{NL} = (\forall x)A^{NL}$.
- $(\neg (\forall x)A)^{NR} = (\exists x)A^{NR}$.

It is straightforward to show that the translations convert NLeft and NRight wffs into their QN-normal forms.

**Definition 20** Define operations $(\cdot)^{DL}$ and $(\cdot)^{DR}$ on QN-normal NLeft and NRight wffs respectively as follows:
For all atoms $p$, $p^D = p^R = p$.

- $(A \& B)^{DL} = A^{DL}\& B^{DL}$
- $(A \rightarrow B)^{DL} = A^{DR} \rightarrow B^{DL}$
- $(\forall_1 A)^{DL} = \forall_1 (A)^{DL}$
- $(\neg (A \vee B))^{DL} = (\neg A)^{DL} \& (\neg B)^{DL}$
- $(\neg \neg A)^{DL} = A^{DL}$
- $(A \oplus B)^{DR} = A^{DR} \oplus B^{DR}$ where $\oplus$ is $\circ$ or $\vee$.
- $(A \rightarrow B)^{DR} = A^{DL} \rightarrow B^{DR}$
- $(\exists_1 A)^{DR} = \exists_1 (A)^{DR}$
- $(\neg (A \oplus B))^{DR} = (\neg A)^{DR} \otimes (\neg B)^{DR}$ where $\oplus$ is $+$ or $\&$ and $\otimes$ is respectively $\circ$ or $\&$.
- $(\neg \neg A)^{DR} = A^{DR}$
- $(\neg (A \rightarrow B))^{DR} = A^{DR} \circ (\neg B)^{DR}$

**Theorem 84** Where $A$ is an NLeft (NRight) wff, $(A^{NL})^{DL}$ $(A^{NR})^{DR}$ is a Q-normal Left (Right) wff.

It is easy to see that NLeft and NRight wffs fulfill the conditions of Th. 73: that is, where $A$ is an NLeft wff and $B$ is an NRight wff, $A^D$ and $B^D$ are such that neither contain any occurrences of $+$ and $\neg$ occurs only in negative positions in $A^D$ and in positive positions in $B^D$. So we can conclude that $B$ is in the smallest $QL^{DTEF}$-theory containing $A$ iff $A^* \vdash B^*$ is provable in $GLR_+$, where $A^*$ and $B^*$ are as in the statement of the theorem. On the other hand, it is straightforward to show that $A^*$ is a Left wff and $B^*$ is a Right wff, and so $A^* \vdash B^*$ is provable in $GLR_+$ iff $(A^*)^A \vdash (B^*)^Q$ is provable in $GLR_{AQ}$. We can also show that $(A^{NL})^{DL} = (A^*)^L$ and $(B^{NR})^{DR} = (B^*)^R$. Since $(A^L)^A = A^A$ for any Left wff
and \((B^R)^Q = B^Q\) for any Right wff, this tells us that \((A^*)^A = ((A^{NL})^{DL})^A\) and \((B^*)^Q = ((B^{NR})^{DR})^Q\). So the final result is:

**Theorem 85** For NLeft \(A\) and NRight \(B\), \(B\) is in the smallest \(\text{QL}^{DTEF}\)-theory containing \(A\) iff \(((A^{NL})^{DL})^A \vdash ((B^{NR})^{DR})^Q\) is provable in \(\text{CLR}_{AQ}\).

It would have been simpler to combine the \((.)^{NL}\) and \((.)^{DL}\) translations into a single operation (similarly for \((.)^{NR}\) and \((.)^{DR}\) but the class of QN-normal forms is important for our logic programming system as it determines the syntax of the system's user-level language: see Sec. 4.6.

Readers should note here how the adoption of the \(F\) principle together with the restriction on NLeft wffs disallowing negated implications as assertions results in a system where we can at least allow them as queries. If we were to allow negated implicational assertions, this would require us to find computational mechanisms for dealing with permission (translating the assertion \(\neg(p \rightarrow q)\) as \(p \rightarrow \neg q\)); as it is the \(F\) principle allows us to replace permission by fusion, because under the restrictions, \(p \rightarrow q\) will be provable only when \(pq\) is (which of course would not be the case if our database were to contain permission statements). Determining whether or not there is a natural and efficient extension to the computation allowing full permission is a moderately interesting avenue for further work.

### 4.4 Conjunction Revisited

In this section we give some brief remarks intended to provide more of a context for the ambiguous `\&` conjunction of our computation. Let us consider just the conjunctive clauses of the L/R restrictions, defining a two new classes of formulas:

- A **Conjunction-Left (CLeft)** wff has no positive occurrences of \(\circ\) and no negative occurrences of \&.

- A **Conjunction-Right (CRight)** wff has no negative occurrences of \(\circ\) and no positive occurrences of \(\circ\).
Clearly, every Left (Right) wff is a CLeft (CRight) wff. A \textit{CL/CR sequent} has the form $\alpha \vdash A$ where $\alpha$ is a multiset of CLLeft wffs and $A$ is a CRight wff. Let $\mathcal{GLR}_C$ be the Gentzen system which results from $\mathcal{GLR}_+$ by dropping the rules $(\circ \vdash)$ and $(\vdash \&)$. It is straightforward to show that there is a $\mathcal{GLR}_C$ proof of a CL/CR sequent iff there is a $\mathcal{GLR}_+$ proof. Now define a new Gentzen system, $\mathcal{GLR}_\Lambda$, by replacing the $(\& \vdash)$ rule by a similar $(\wedge \vdash)$ rule in which $\wedge$ replaces $\&$, replacing the $(\vdash \circ)$ rule by a $(\vdash \wedge)$ rule which is the same except that $\wedge$ replaces $\circ$. Again it is straightforward to show that there is a $\mathcal{GLR}_C$ proof of a CL/CR sequent $\alpha \vdash A$ iff there is a $\mathcal{GLR}_\Lambda$ proof of $\alpha^\wedge \vdash A^\wedge$.

The most notable feature of $\mathcal{GLR}_\Lambda$ is that \textit{Cut} is not admissible for it. Both of the following are provable in $\mathcal{GLR}_+$:

\begin{align*}
A &\vdash A \circ (A \& B \rightarrow A) \\
A \& (A \circ B \rightarrow A) &\vdash B \rightarrow A
\end{align*}

but this is not:

\begin{align*}
A &\vdash B \rightarrow A
\end{align*}

(If it were, $A \rightarrow . B \rightarrow A$ would be a theorem of $\mathcal{LR}_+$. ) Now all of these sequents are CL/CR sequents, so in $\mathcal{GLR}_\Lambda$ the following are provable:

\begin{align*}
A &\vdash A \wedge (A \wedge B \rightarrow A) \\
A \wedge (A \wedge B \rightarrow A) &\vdash B \rightarrow A
\end{align*}

but again:

\begin{align*}
A &\vdash B \rightarrow A
\end{align*}

is not. In the translation, the right hand side of the first sequent and the left hand side of the second are identical, and so we have demonstrated the failure of \textit{Cut} for $\mathcal{GLR}_\Lambda$. This explains the peculiarity of the computation that we noted at the
CHAPTER 4. RELEVANT LOGIC PROGRAMMING

beginning of Sec. 4.3; the failure of Modus Ponens for the translated system is of course explained by the fact that our univocal \( \land \) conjunction has different logical properties depending on its syntactic position. For a logic programming system we feel that this is not an important enough problem to force the abandonment of \( \land \) in favour of explicit \& and \( \circ \); however the possibility remains open.

If we compare \( QLR_\land \) to the positive part of the intuitionistic Gentzen system \( LJ \) of [Sza69, III.1], we see that the only essential difference is that \( QLR_\land \) does not have the Weakening rule.

\[
(K \vdash) \frac{\alpha \vdash A}{\alpha, B \vdash A}
\]

\( LJ \) as it is formulated there includes an explicit \textit{Cut} rule; Gentzen's \textit{Hauptsatz} of [Sza69, III.1] shows that this can be dropped from the system without depleting the stock of provable sequents. In \( LJ \) the sequents are formed with lists rather than the multisets on the left, so the system also includes an Interchange rule allowing the permutation of elements (which is of course unnecessary in a multiset formulation). So we could restore \textit{Cut} to the translated system by adding \((K \vdash)\); the result would be Gabbay and Reyle's NPROLOG. For a relevant system however, \((K \vdash)\) is clearly invalid, given the interpretation of a sequent \( \alpha \vdash A \) as asserting the provability of \( A \) from \( \alpha \) in a way which uses every element of \( \alpha \).

On the other hand, \((K \vdash)\) becomes admissible if we add an explicit \textit{Cut} rule. \( A \land B \vdash A \) is provable in \( GLR_\land \) (translating as \( A \& B \vdash A \)), and \( \alpha, B \vdash A \land B \) follows from \( \alpha \vdash B \) (by an application of \((\vdash \land)\), using \( B \vdash B \) as the other premiss). \( \alpha, B \vdash A \) follows by an application of \textit{Cut}. So for our translated system, the conclusion is that \textit{Cut} would be admissible if and only if the system were irrelevant.

Our ambiguous reading of \( \land \) has a close though not very well explored relationship with independent work by Tennant on 'Classical Relevant Logic' ([Ten84]). Conjunction in this system corresponds to our \( \land \) (though it is taken as primitive there, rather than 'cooked up' out of \( \circ \) and \&), and the failure of \textit{Cut} is motivated largely by the desire to retain Disjunctive Syllogism while avoiding \textit{ex falso}. 
4.5 Comparing the Logic and the Computation

4.5.1 Overview

Obviously there is a rough correspondence between computation tree nodes and \( A/Q \) sequents, and between computation trees and \( \mathcal{GLR}_{AQ} \) proofs. The sequence \( (S_1 \ldots S_m) \) in a node corresponds to the left-hand side of the sequent and the query \( A \) to the right-hand side. \(^3\) The \( \mathcal{GLR}_{AQ} \) rules \((\vdash \circ), (\vdash \vee), (\rightarrow \vdash)\) and \((\vdash \rightarrow)\) have their counterparts in the Fuse, Vel, Chain and Sub rules, and the non-deterministic candidate selection is expressed by the \((\& \vdash)\) rules. The most obvious difference is the book-keeping and associated work involved in generating the answer substitution in the computational model. More important, however, is the use of \( D_1 \) and \( D_0 \) to record which of the segments \( S_1 \ldots S_m \) have been used in the computation.

If we examine the \((\vdash \circ)\) rule, as a first cut it suggests a computational strategy whereby in attempting to prove \( \alpha \vdash A \circ B \) we consider all the decompositions of \( \alpha \) into \( \beta \) and \( \gamma \) s.t. \( \alpha = \beta, \gamma \), and try to prove \( \beta \vdash A \) and \( \gamma \vdash B \) for each of these in turn until we succeed (we will also have to consider the effects of the \((W \vdash)\) rule - see below). In a working system we would probably abandon this naive generate and test method in favour of a procedure which limits the number of decompositions - maybe this: given \( \alpha \vdash A \circ B \) to prove, find \( \alpha_1 \) and \( \alpha_2 \) (both weak submultisets of \( \alpha \)) for which \( \alpha_1 \vdash A \) and \( \alpha_2 \vdash B \) are provable, with backtracking induced by a check that \( \alpha = \alpha_1, \alpha_2 \). The idea then is to carry all of \( \alpha \) up the tree, and indicate \( \alpha_1 \) and \( \alpha_2 \) by some mask on \( \alpha \) on each branch, building them up according to which elements of \( \alpha \) are actually used on the branch. We now have a system which, given a sequent \( \alpha \vdash A \), will construct a weak submultiset \( \alpha' \) of \( \alpha \) for which \( \alpha' \vdash A \) is provable (if there is one).

In a full implementation of such a system the issue of what checks to perform and when to perform them is nontrivial: we have to determine the points at which

\(^3\)The biggest difference here is that in the computational description the left hand side is a list, whereas sequents have multisets - but as we point out on p.188, the database segments in a computation tree node may be permuted freely.
it becomes necessary for particular elements of \( \alpha \) to have been used. This is essentially a description of the behaviour of computation trees. For our purposes, the only 'use check' necessary is the one performed in the Sub rule, essentially because the only way new segments get generated is through applications of the corresponding rule \((\vdash \rightarrow)\). The result in \( \mathcal{GLR}_{AQ} \) terms is spelled out in the Soundness theorem below.

A more subtle point is raised by the absence of any specific computation tree counterpart to the contraction rule \((W \vdash)\). In fact we could have formulated \( \mathcal{GLR}_{AQ} \) without \((W \vdash)\), by building contraction into the other rules. Where the left-hand side of the conclusion of a rules was originally some multiset \( \alpha \), we could have made it \([\alpha]\), where \([\alpha]\) stands for any strong sub-multiset of \( \alpha \).

We get the same effect in the Chain and Fuse rules, by taking set-theoretical unions in constructing \( D_\rho \). While on the issue of contraction, note that if we did not have \((W \vdash)\) as a rule of \( \mathcal{GLR}_{AQ} \) then this simple manipulation of masks as sets would not work. In particular, our 'use check' would not be just that the appropriate segment was used; we would have to check that the segment was used exactly once, presumably making use of masks which were multisets rather than sets, and leading to a more expensive computation. This is of some interest when comparing the logical basis for this approach with Girard's Linear Logic ([Gir87]), as \( \mathcal{GLR}_{AQ} \) is a fragment of Linear Logic plus \((W \vdash)\) (under the appropriate translation of the symbols used for the connectives - see [Rez87] and [Avr88]).

### 4.5.2 Substitutions

We pause to give some results for substitutions which will be needed for the soundness and completeness proofs.

**Definition 21 (Linear Substitutions)** A substitution \( \theta \) is linear iff \( t\theta = t \) for all \( t/x \in \theta \).
Such a substitution \( \theta \) is the composition of all its elements, in any order, and so \( A\theta \) can be obtained by successively applying each element of \( \theta \) to \( A \).

**Theorem 86** A substitution \( \theta \) is linear iff \( \theta \theta = \theta \).

**Proof**

- Let \( \theta \) be linear. If \( t/x \in \theta \) then \( x\theta /x \in \theta \). But \( x\theta = t\theta = t \). So \( t/x \in \theta \).
  
  If \( t/x \in \theta \theta \) then \( t = x\theta \) for some \( s/x \in \theta \). But then \( t = s\theta = s \), so \( t/x \in \theta \).

- Let \( \theta \theta = \theta \). For any \( t/x \in \theta \), \( t = x\theta = t\theta \).

As a corollary, if \( \theta \) is the mgu of some set then \( \theta \) is linear, since \( \theta \theta = \theta \) (by the definition of mgu).

**Definition 22 (Weakening)** \( \theta \) weakens \( \rho \) iff \( \rho \theta \rho = \rho \).

**Theorem 87** Let \( \theta \) and \( \rho \) be substitutions s.t. \( \theta \) weakens \( \rho \). Then:

- If \( \rho \theta = \theta \) then \( \theta \rho = \theta \).

- If \( \theta \) is linear then \( \rho \theta \) is linear.

- If \( \theta > \sigma \) then \( \rho \theta > \rho \sigma \).

**Theorem 88** If \( x \) does not occur in \( \theta \) for all \( s/x \in \rho \) then \( \theta \rho = \theta \cup \rho \).

**Proof** By definition, \( \theta \rho = \{x\theta \rho/x : s/x \in \rho \} \). If \( s/x \in \theta \) then \( x\theta \rho = s\rho = s \). If \( s/x \in \rho \) then \( x\theta \rho = x\rho = s \).

**Theorem 89** If \( \theta \cup \rho \) is a substitution and \( \rho \) is linear then \( \rho (\theta \cup \rho) = \rho \theta \).

**Proof** \( \rho (\theta \cup \rho) = \{x\rho (\theta \cup \rho)/x : s/x \in \rho \cup \theta \cup \rho = \theta \cup \rho \} \). But \( x\rho (\theta \cup \rho) = x\rho \) (\( X\rho = X \rho \) since \( \rho \) is linear).

As a corollary, If \( \theta \rho = \theta \cup \rho \) and \( \rho \) is linear then \( \theta \) weakens \( \rho \).
Theorem 90  If \( \rho \) is linear and \( \theta \) is the mgu of some set of elements \( \{A_1\rho \ldots A_m\rho\} \) then \( \theta \) weakens \( \rho \).

Proof  This will follow by Th. 89 if we can show that \( \theta \rho = \theta \cup \rho \), which in turn follows by Th. 88 if we can show that \( x \) does not occur in \( \theta \) for all \( s/x \in \rho \). So let \( s/x \in \rho \). Since \( \rho \) is linear, \( A_i\rho \rho = A_i\rho \) for all the \( A_i \) and so \( x \) does not occur in any \( A\rho \) (note that \( s = x \) is not possible, by the definition of a substitution in Sec. 4.2.2). So clearly \( x \) will not occur in \( \theta \).

We make a minor additional stipulation to the computation tree conditions: for each node, the substitutions \( \theta_I \) and \( \theta_O \) must be linear. It is not hard to see that where the \( \theta_I \) substitution in the root node is linear, all substitutions in the tree will be. For every substitution in the tree is either \( \theta_I \) or of the form \( \theta_I\theta_2 \) where \( \theta_2 \) is the mgu of a set of elements of the form \( \{A_1\theta_1 \ldots A_m\theta_1\} \). If we have as inductive hypothesis the supposition that \( \theta_1 \) is linear, then by the previous theorem \( \theta_2 \) weakens \( \theta_1 \) and so \( \theta_1\theta_2\theta_1\theta_2 = \theta_1\theta_2\theta_2 = \theta_1\theta_2 \) (since \( \theta_2 \) is linear, being an mgu).

4.5.3  Soundness of the Computation

For convenience we will introduce the following ‘masking’ operation:

\[
\{k_1 \ldots k_n\}/\langle S_1 \ldots S_m \rangle = [S_{k_1} \ldots S_{k_n}] 
\]

where \( \{k_1 \ldots k_n\} \subseteq \{1 \ldots m\} \). In other words, \( D/\langle S_1 \ldots S_m \rangle \) constructs a multiset having one occurrence of \( S_i \) for each \( i \in D \).

We will also need the following lemma, whose proof is straightforward:

Lemma 8  If \( S_1 \ldots S_m \vdash A \) is provable in \( GLR_{AQ} \) then so is \( S_1\theta \ldots S_m\theta \vdash A\theta \) for any substitution \( \theta \).

Theorem 91 (Soundness)  Let \( S \) be an assertion and \( A \) a query. If

\[\text{Proof}( (\forall S') , A^\wedge , D , \{1\} , \varepsilon , \theta) \]
(where \( D \) is \( \{1\} \) if \( t \in S \) and \( \emptyset \) otherwise) then there is a \( \mathcal{GLR}_{AQ} \) proof of \( \forall S \vdash A \theta \).

**Proof** We will show that if

\[
\text{Proof}(\langle S^1 \ldots S^m \rangle, A^\top, D_I, D_O, \theta_I, \theta_O)
\]

then there is a \( D \) for which \( \emptyset \subseteq D \) and \( D_O - D_I \subseteq D \subseteq D_O \), such that there is a \( \mathcal{GLR}_{AQ} \) proof of

\[
D/(S_1 \theta_O \ldots S_m \theta_O) \vdash A \theta_O
\]

The theorem will follow immediately. Proof by induction on the computation tree.

- Consider a tip node

\[
\langle S^1 \ldots S^m \rangle, p, D, D \cup \{i\}, \theta_I, \theta_I \theta
\]

where for some \( S_i \),

- \( S_i = \forall_i A_i \)
- \( p' \in A_i^\top \)
- \( p \) is a \( \forall_i \)-renaming.
- \( p \theta_I \) and \( p' \rho \theta_I \) unify with mgu \( \theta \).

\[
p' \rho \theta_I \theta \vdash p \theta_I \theta
\]

is an instance of \( (Id) \).

\[
A_i \theta_I \theta \vdash \rho \theta_I \theta
\]

follows by repeated applications of \( (\& \vdash) \), and

\[
\forall_i S_i \theta_I \theta \vdash p \theta_I \theta
\]
by applications of $\forall \vdash \cdot$. So clearly $\{i\}$ has the required properties.

- Show that the statement holds for a node (other than a Tip node) under the assumption that it holds for the node's children.

**Ad Fuse** Consider a node

$$\langle (S_1^\wedge \ldots S_m^\wedge), (A \circ B)^\wedge, D_I, D_O, \theta_I, \theta_O \rangle$$

with children

$$\langle (S_1^\wedge \ldots S_m^\wedge), A^\wedge, D_I, D, \theta_I, \theta \rangle$$

and

$$\langle (S_1^\wedge \ldots S_m^\wedge), B^\wedge, D, D_O, \theta, \theta_O \rangle$$

By the inductive hypothesis we have

$$D_1/(S_1 \theta \ldots S_m \theta) \vdash A \theta$$

and

$$D_2/(S_1 \theta \ldots S_m \theta) \vdash B \theta_O$$

for $\emptyset \subset D_1, D_2$ and $D = D_I \subset D_1 \subset D$ and $D_O - D \subset D_2 \subset D_O$. By the lemma,

$$D_1/(S_1 \theta \ldots S_m \theta) \vdash A \theta_O$$

(since $\theta \theta_O$). Let $D' = D_1 \cup D_2$.

$$D'/(S_1 \theta \ldots S_m \theta) \vdash (A \circ B) \theta_O$$

follows by ($\vdash \circ$) and possible steps of $(W \vdash)$, and it is easy to see that $D'$ has the required properties.

**Ad Vel** Straightforward using the inductive hypothesis and an application of ($\vdash \lor$).

**Ad Chain** Consider a node

$$\langle (S_1^\wedge \ldots S_m^\wedge), p, D_I, D_O, \theta_I, \theta_O \rangle$$

with child
\[(\{S^1_1 \ldots S^1_m\}, B^0 \rho, D_I, D, \theta_I, \theta_O)\]

s.t. for some \(S_i,\)
- \(S_i = \forall_i A_i\)
- \(B \rightarrow p' \in A_i.\)
- \(\rho\) is a \(\forall_i\)-renaming.
- \(p\theta_I\) and \(p'\rho\theta_I\) unify with mgu \(\theta.\)
- \(D_O = D \cup \{i\}.\)

the inductive hypothesis,
\[D'/(S_1 \theta_O \ldots S_m \theta_O) \vdash B \rho \theta_O\]
for some \(\emptyset \subset D',\) \(D - D_I \subset D' \subset D\) and by an application of \((\rightarrow p \vdash),\)
\[D'/(S_1 \theta_O \ldots S_m \theta_O), (A \rightarrow p') \rho \theta_O \vdash p' \rho \theta_O\]

By applications of \((\forall \vdash)\) and \((\& \vdash)\) we get
\[D'/(S_1 \theta_O \ldots S_m \theta_O), S_i \theta_O \vdash p' \rho \theta_O\]

and with a \((W \vdash)\) step we get
\[(D' \cup \{i\})/(S_1 \theta_O \ldots S_m \theta_O) \vdash p' \rho \theta_O\]

Clearly \(p' \rho \theta_O = p\rho \theta_O,\) and \(D' \cup \{i\}\) has the required properties.

**Ad Sub** Consider a node
\[\((\{S^1 \ldots S^m_1\}, (A \rightarrow B)^\wedge, D_I, D_O, \theta_I, \theta_O)\]

with child
\[\((\{S^1 \ldots S^m_1\}, B^\wedge, D_I', D, \theta_I, \theta_O)\]

where \(m + 1 \in D\) and \(D_O = D - \{m + 1\},\) and \(D_I'\) is either \(D_I\) or \(D_I \cup \{m + 1\}\) depending on whether or not \(t \in A.\) By the inductive hypothesis, there is a \(\emptyset \subset D',\) \(D - D_I' \subset D' \subset D\) s.t.
\[D'/(S_1 \theta_O \ldots S_m \theta_O, A \theta_O) \vdash B \theta_O\]

If \(D_I' = D_I\) then *a fortiori* \(m + 1 \in D'\); but we can always assume that \(m + 1 \in D'\) because
CHAPTER 4. RELEVANT LOGIC PROGRAMMING

\[ D'/(S_1\theta_O \ldots S_m\theta_O), A\theta_O \vdash B\theta_O \]

follows by applications of \((t \vdash), (\& \vdash)\) and \((\forall \vdash)\), assuming that \(t \in A\).

By an application of \((\vdash \rightarrow)\),

\[ (D' - \{m + 1\})/(S_1\theta_O \ldots S_m\theta_O) \vdash (A \rightarrow B)\theta_O \]

and the result follows by noting that \(D' - \{m + 1\}\) has the required properties.

As a corollary, we have the following:

**Theorem 92** Let \(A\) be an unquantified assertion and \(G\) an unquantified query.

If

\[
\text{Proof}( (\forall A^\wedge), G^\wedge, D, \{1\}, \epsilon, \theta) 
\]

(where \(D\) is \(\emptyset\) if \(t \notin A\) and \(\{1\}\) otherwise) then there is a \(\mathcal{GLR}_+\) proof of

\[ \forall S \vdash \forall A\theta \]

**Proof** Under the hypotheses we know by the previous theorem that there is a \(\mathcal{GLR}_{AQ}\) proof, and so a \(\mathcal{GLR}_+\) proof, of \(\forall S \vdash A\theta\), and the result follows by applications of \((\vdash \forall)\).

So we are permitted, as in normal PROLOG, to interpret free variables in the answer universally.

### 4.5.4 Completeness of the Computation

The completeness proof will use the following properties of computation trees.

**Permutation** If

\[
\text{Proof}( (S_1 \ldots S_m), A, D_I, D_O, \theta_I, \theta_O) 
\]
and \( \langle S_{k_1} \ldots S_{km} \rangle \) is some permutation of \( \langle S_1 \ldots S_m \rangle \) and \( D'_I = \{ i : k_i \in D_I \} \), and \( D'_O = \{ i : k_i \in D_O \} \) then

\[
\text{Proof}(\langle S_{k_1} \ldots S_{km} \rangle, A, D'_I, D'_O, \theta_I, \theta_O)
\]

**Expansion If**

\[
\text{Proof}(\langle S_1 \ldots S_m \rangle, A, D_I, D_O, \theta_I, \theta_O)
\]

then

\[
\text{Proof}(\langle S_1 \ldots S_m, S_{m+1} \rangle, A, D_I, D_O, \theta_I, \theta_O)
\]

for any \( S_{m+1} \).

**Deletion If**

\[
\text{Proof}(\langle S_1 \ldots S_{m+1} \rangle, A, D_I, D_O, \theta_I, \theta_O)
\]

and \( m + 1 \notin D_O \) then

\[
\text{Proof}(\langle S_1 \ldots S_m \rangle, A, D_I, D_O, \theta_I, \theta_O)
\]

**Contraction If**

\[
\text{Proof}(\langle S_1 \ldots S_{m+1} \rangle, A, D_I, D_O, \theta_I, \theta_O)
\]

where \( S_m = S_{m+1} \) and \( m + 1 \in D_O \) then

\[
\text{Proof}(\langle S_1 \ldots S_{m+1} \rangle, A, D', D_O \cup \{m\} - \{m + 1\}, \theta_I, \theta_O)
\]

where if \( m + 1 \in D_I \) then \( D' = D_I \cup \{m\} - \{m + 1\} \) else \( D' = D_I \).

**\( t \)-Weakening If**

\[
\text{Proof}(\langle S_1 \ldots S_m \rangle, A, D_I, D_O, \theta_I, \theta_O)
\]

then
CHAPTER 4. RELEVANT LOGIC PROGRAMMING

\[ Proof((S_1 \ldots S_m, t), A, D_I, D_O \cup \{m + 1\}, \theta_I, \theta_O) \]

\&-Weakening If

\[ Proof((S_1 \ldots S_m), A, D_I, D_O, \theta_I, \theta_O) \]

then

\[ Proof((S_1 \ldots S'_m), D_I, D_O, \theta_I, \theta_O) \]

where \( S_m = \forall_m A \) and \( S'_m = \forall_m (A \land B) \) for any unquantified assertion \( B \).

\( \theta \)-Weakening If

\[ Proof((S_1 \ldots S_m), A, D_I, D_O, \theta_I, \theta_O) \]

and \( \tau \) is a linear substitution composed of elements of the form \( Y \setminus t \) where \( Y \) occurs nowhere in the computation tree then

\[ Proof((S_1 \ldots S_m), A, D_I, D_O, \tau \theta_I, \tau \theta_O) \]

\( \theta \)-Strengthening If

\[ Proof((S_1 \ldots S_m), A, D_I, D_O, \theta_I, \theta_O) \]

and \( \theta'_I \theta_I \) for some linear \( \theta'_I \) then

\[ Proof((S_1 \ldots S_m), A, D_I, D_O, \theta'_I, \theta'_O) \]

for some \( \theta'_O > \theta_O \).

Query Equivalence If

\[ Proof((S_1 \ldots S_m), A, D_I, D_O, \theta_I, \theta_O) \]

and \( B \theta_I = A \theta_I \) then

\[ Proof((S_1 \ldots S_m), B, D_I, D_O, \theta_I, \theta_O) \]
\section*{CHAPTER 4. RELEVANT LOGIC PROGRAMMING}

\textbf{θ-Shift} If

\[ \text{Proof}(\langle S_1 \theta \ldots S_m \theta \rangle, A \theta, D_I, D_O, \theta I, \theta O) \]

then

\[ \text{Proof}(\langle S_1 \ldots S_m \rangle, A, D_I, D_O, \theta I, \theta O) \]

\textbf{D-Weakening} If

\[ \text{Proof}(\langle S_1 \ldots S_m \rangle, A, D_I, D_O, \theta I, \theta O) \]

and \( X \subseteq \{1 \ldots m\} \) then

\[ \text{Proof}(\langle S_1 \ldots S_m \rangle, A, D_I \cup X, D_O \cup X, \theta I, \theta O) \]

Here we will only give a proof of the Contraction property. We will show that if

\[ \text{Proof}(\langle S_1 \ldots S_m \rangle, A, D_I, D_O, \theta I, \theta O) \]

and for some \( i \neq j \), \( S_i = S_j \) and \( i \in D_O \) then

\[ \text{Proof}(\langle S_1 \ldots S_m \rangle, A, D', D_O \cup \{j\} - \{i\}, \theta I, \theta O) \]

where if \( i \in D_I \) then \( D' = D_I \cup \{j\} - \{i\} \) else \( D' = D_I \). The result will follow by Permutation. The proof proceeds by induction on the computation tree:

- The statement is clearly true for any Tip node.

- Consider a node \( (\langle S_1 \ldots S_m \rangle, A, D_I, D_O, \theta I, \theta O) \) in the tree where \( S_i = S_j, i \neq j \) and \( i \in D_O \). Show that the statement holds for this node, under the assumption that it holds for the node’s children.

\textbf{Ad Fuse} If \( A = B \land C \) then by the Fuse rule it has children

\[ (\langle S_1 \ldots S_m \rangle, B, D_I, D, \theta I, \theta) \]

and
is a valid tip node.

- Consider a line in the proof $S_1 \ldots S_m \vdash A$. Show that the statement holds for this line, under the assumption that it holds for the line’s children.

**Ad** ($\to p \vdash$) If the line follows by ($\to p \vdash$) then $A = p$ for some atom $p$ (other than $t$), and for some $1 \leq i \leq m$, $S_i = B^\to p$, for some $B$. We can assume that $i = m$, without loss of generality. The line follows from

$$S_1 \ldots S_{m-1} \vdash B$$

and by the inductive hypothesis,

$$Proof((S_1^{\land} \ldots S_{m-1}^{\land}), B^{\land}, D, \{1 \ldots m-1\}, \epsilon, \theta)$$

By Expansion,

$$Proof((S_1^{\land} \ldots S_{m-1}^{\land}, B^{\land} \to p), B^{\land}, D, \{1 \ldots m-1\}, \epsilon, \theta)$$

Finally the Chain rule gives

$$Proof((S_1^{\land} \ldots S_{m-1}^{\land}, B^{\land} \to p), p, D, \{1 \ldots m\}, \epsilon, \theta)$$

**Ad** ($\vdash \to$) If the line follows by ($\vdash \to$) then $A = B \to C$ and the line follows from

$$S_1 \ldots S_m, B \vdash C$$

By the inductive hypothesis,

$$Proof((S_1^{\land} \ldots S_m^{\land}, B^{\land}), C^{\land}, D', \{1 \ldots m+1\}, \epsilon, \theta)$$

and so by the Sub rule

$$Proof((S_1^{\land} \ldots S_m^{\land}, (B \to C)^\land), D, \{1 \ldots m\}, \epsilon, \theta)$$

where $D' = D$ if $t \notin B$ and $D' = D \cup \{m+1\}$ otherwise.

**Ad** ($\& \vdash$) If the line follows by ($\& \vdash$) then the result is straightforward, by the inductive hypothesis and the $\&$-Weakening property.

**Ad** ($\vdash o$) Let the line follow by an application of ($\vdash o$):

$$S_1, \ldots, S_k \vdash B \quad S_{k+1}, \ldots, S_m \vdash C$$

$$\frac{}{S_1, \ldots, S_m \vdash B \& C}$$
By the inductive hypothesis
\[ \text{Proof}\left((S^1 \cdots S^k), B^\wedge, D_1, \{1 \ldots k\}, \epsilon, \theta_1\right) \]

and
\[ \text{Proof}\left((S^1_{k+1} \cdots S^m_\wedge), C^\wedge, D_2, \{k + 1 \ldots m\}, \epsilon, \theta_2\right) \]

By Expansion and D-Weakening,
\[ \text{Proof}\left((S^1 \cdots S^m_\wedge), B^\wedge, D_1 \cup D_2, \{1 \ldots k\} \cup D_2, \epsilon, \theta_1\right) \]

and by Expansion, Permutation, \(\theta\)-Weakening and D-Weakening,
\[ \text{Proof}\left((S^1 \cdots S^m_\wedge), C^\wedge, \{1 \ldots k\} \cup D_2, \{1 \ldots m\}, \theta_1, \theta_1 \theta_2\right) \]

Finally, the Fuse rule gives
\[ \text{Proof}\left((S^1 \cdots S^m_\wedge), (B o C)^\wedge, D_1 \cup D_2, \{1 \ldots m\}, \epsilon, \theta_1 \theta_2\right) \]

and clearly \(\theta_1 \theta_2\) has the correct form.

\textbf{Ad} (\(\vdash \lor\)) Straightforward, by the inductive hypothesis and the Vel rule.

\textbf{Ad} (\(\forall \vdash\)) Let the line follow by an application of (\(\forall \vdash\)):

\[ S_1, \ldots, S_m, B\{s/x\} \vdash C \]

\[ S_1, \ldots, S_m, (\forall x)B \vdash C \]

By the inductive hypothesis,
\[ \text{Proof}\left((S^1_\wedge \cdots S^m_\wedge, B^\wedge\{s/x\}), C^\wedge, D, \{1 \ldots m + 1\}, \epsilon, \theta\right) \]

By Th. 93,
\[ \text{Proof}\left((S^1_\wedge \cdots S^m_\wedge, (\forall x)B^\wedge), C^\wedge, D, \{1 \ldots m + 1\}, \epsilon, \theta'\right) \]

where \(\theta' > \tau \theta\), where \(\tau\) is composed of elements of the form \(t/x\) where \(x'\) is a renaming of \(x\) in accordance with the Tip and Chain rules. Clearly \(\tau \theta\) has the correct form, and a fortiori \(\theta'\) must also, it being a trivial matter to show that for any \(u/y \in \theta'\), \(t/y \in \tau \theta\) for some \(t\).

\textbf{Ad} (\(W \vdash\)) By the Contraction and Deletion properties.

\textbf{Ad} (\(t \vdash\)) By the \(t\)-Weakening property.
Theorem 95 (Completeness) Let $S$ be an assertion and let $A$ be an unquantified query. If there is a $\mathcal{GLR}_{AQ}$ proof of $\forall S \vdash \exists A$ then the set $\Theta$ of substitutions $\theta$ for which $\forall S \vdash A\theta$ is provable is non-empty, and for each such $\theta$ there is a $\theta' \in \Theta$ such that

- $\text{Proof}((\forall S^\wedge), A^\wedge, D, \{1\}, \epsilon, \theta')$ (where $D$ is either $\{1\}$ or $\emptyset$ depending on whether or not $t \in S$)

- $A\theta = A\theta'\sigma$ for some $\sigma$ (so $A\theta'$ is more general than $A\theta$).

Proof The existence of $\Theta$ follows by Th. 78. By Th. 94, for each $\theta \in \Theta$ we have

$$\text{Proof}((\forall S^\wedge), A^\wedge, D, \{1\}, \epsilon, \rho)$$

where $\rho$ is composed of elements of the form $t/x$ where $x$ is a renaming in accordance with the Tip and Chain rules. Since $\forall S^\wedge = (\forall S^\wedge)\theta$, by $\theta$-Shift we know

$$\text{Proof}((\forall S^\wedge), A^\wedge, D, \{1\}, \theta, \theta'\rho)$$

and by $\theta$-Strengthening,

$$\text{Proof}((\forall S^\wedge), A^\wedge, D, \{1\}, \epsilon, \theta')$$

for some $\theta' > \theta\rho$. That $\theta' \in \Theta$ follows from the Soundness theorem. Let $\sigma = \theta\rho$. We know that $A\theta'\sigma = A\sigma$. Further, $A\sigma = A\theta$, since clearly none of the renaming variables substituted for by $\rho$ can occur in $A\theta$.

4.6 Implementation

The relevant extension to (positive) PROLOG presented here in its CQEP guise has been implemented as part of the experimental CLOGPROG system, using a simple extension to the simple stack-frame, structure sharing PROLOG interpreter as described in [Hog84].
At the user-level, assertions are of the form $A'$, where $A$ is an unquantified NLeft wff (that is, a QN-normal Left wff with no leading quantifiers), and $A'$ results from $A$ by replacing all occurrences of $\&$ and $\circ$ by $\land$, and all occurrences of $+$ and $\lor$ by a single disjunction sign (in fact we use the PROLOG-standard ';'). The operations performed by the parser are implicitly as follows:

- Convert $A'$ to $A$.
- Convert $A$ to the unquantified Left wff $B = A^{DL}$ by moving negation signs in until they occur only in atomic contexts.
- Convert $B$ to the unquantified assertion $B_1 \& \ldots \& B_m = B^A$.
- Add each elementary assertion $B_i$ to the database.

User-level queries are of the form $A'$, where $A$ is an unquantified NRight wff, $A'$ results from $A$ again by replacing all occurrences of $\&$ and $\circ$ by $\land$, and all occurrences of $+$ and $\lor$ by ';'. The parsing operations are as follows:

- Convert $A'$ to $A$.
- Convert $A$ to the unquantified Right wff $B = A^{DR}$.
- Convert $B$ to the unquantified query $B_1 \circ \ldots \circ B_m = B^Q$, and form the existential closure.

For efficiency reasons, the current implementation does not allow $t$ to appear in assertion positions. Explicit internal universal quantifiers are allowed, as defined in the syntax for QN-normal formulas.

A switch RELEVANT/NORELEVANT controls whether implications are to be interpreted relevantly or intuitionistically, by enabling or disabling the use checks in CQEP. As pointed out above, disabling these gives a system essentially equivalent to the NPROLOG of [GR84]. Setting NORELEVANT is logically equivalent to adding $t$ to every new database segment formed in the course of the computation.
CHAPTER 4. RELEVANT LOGIC PROGRAMMING

Proof By induction on the $\mathcal{GLR}_{AQ}$ proof.

- Consider an application of (Id):

$$p \vdash p$$

Clearly

$$((p), p, D, \{1\}, \epsilon, \epsilon)$$

is a valid tip node.

- Consider a line in the proof $S_1 \ldots S_m \vdash A$. Show that the statement holds for this line, under the assumption that it holds for the line’s children.

\textbf{Ad} ($\rightarrow p \vdash$) If the line follows by ($\rightarrow p \vdash$) then $A = p$ for some atom $p$ (other than $t$), and for some $1 \leq i \leq m$, $S_i = B^\rightarrow p$, for some $B$. We can assume that $i = m$, without loss of generality. The line follows from

$$S_1 \ldots S_{m-1} \vdash B$$

and by the inductive hypothesis,

$$\text{Proof}((S_1^\wedge \ldots S_{m-1}^\wedge), B^\wedge, D, \{1 \ldots m - 1\}, \epsilon, \theta)$$

By Expansion,

$$\text{Proof}((S_1^\wedge \ldots S_{m-1}^\wedge, B^\wedge \rightarrow p), B^\wedge, D, \{1 \ldots m - 1\}, \epsilon, \theta)$$

Finally the Chain rule gives

$$\text{Proof}((S_1^\wedge \ldots S_{m-1}^\wedge, B^\wedge \rightarrow p), p, D, \{1 \ldots m\}, \epsilon, \theta)$$

\textbf{Ad} ($\vdash \rightarrow$) If the line follows by ($\vdash \rightarrow$) then $A = B \rightarrow C$ and the line follows from

$$S_1 \ldots S_m, B \vdash C$$

By the inductive hypothesis,

$$\text{Proof}((S_1^\wedge \ldots S_m^\wedge, B^\wedge), C^\wedge, D', \{1 \ldots m + 1\}, \epsilon, \theta)$$
and so by the Sub rule

\[ \text{Proof}(\langle S_1^\land \ldots S_m^\land, (B \rightarrow C)^\land, D, \{1 \ldots m\}, \varepsilon, \theta \rangle) \]

where \( D' = D \) if \( t \notin B \) and \( D' = D \cup \{m + 1\} \) otherwise.

\textbf{Ad} (\& \vdash) If the line follows by (\& \vdash) then the result is straightforward, by the inductive hypothesis and the \&-Weakening property.

\textbf{Ad} (\vdash \circ) Let the line follow by an application of (\vdash \circ):

\[
\begin{align*}
S_1, \ldots, S_k & \vdash B \\
S_{k+1}, \ldots, S_m & \vdash C
\end{align*}
\]

\[ S_1, \ldots, S_m \vdash B \circ C \]

By the inductive hypothesis

\[ \text{Proof}(\langle S_1^\land \ldots S_k^\land \rangle, B^\land, D_1, \{1 \ldots k\}, \varepsilon, \theta_1) \]

and

\[ \text{Proof}(\langle S_{k+1}^\land \ldots S_m^\land \rangle, C^\land, D_2, \{k + 1 \ldots m\}, \varepsilon, \theta_2) \]

By Expansion and D-Weakening,

\[ \text{Proof}(\langle S_1^\land \ldots S_m^\land \rangle, B^\land, D_1 \cup D_2, \{1 \ldots k\} \cup D_2, \varepsilon, \theta_1) \]

and by Expansion, Permutation, \( \theta \)-Weakening and D-Weakening,

\[ \text{Proof}(\langle S_1^\land \ldots S_m^\land \rangle, C^\land, \{1 \ldots k\} \cup D_2, \{1 \ldots m\}, \theta_1, \theta_1 \theta_2) \]

Finally, the Fuse rule gives

\[ \text{Proof}(\langle S_1^\land \ldots S_m^\land \rangle, (B \circ C)^\land, D_1 \cup D_2, \{1 \ldots m\}, \varepsilon, \theta_1 \theta_2) \]

and clearly \( \theta_1 \theta_2 \) has the correct form.

\textbf{Ad} (\vdash \lor) Straightforward, by the inductive hypothesis and the Vel rule.

\textbf{Ad} (\forall \vdash) Let the line follow by an application of (\forall \vdash):

\[
\begin{align*}
S_1, \ldots, S_m, B\{s/x\} & \vdash C \\
S_1, \ldots, S_m, (\forall x)B & \vdash C
\end{align*}
\]

By the inductive hypothesis,

\[ \text{Proof}(\langle S_1^\land \ldots S_m^\land, B^\land\{s/x\} \rangle, C^\land, D, \{1 \ldots m + 1\}, \varepsilon, \theta) \]

By Th. 93,

\[ \text{Proof}(\langle S_1^\land \ldots S_m^\land, (\forall x)B^\land \rangle, C^\land, D, \{1 \ldots m + 1\}, \varepsilon, \theta') \]
CHAPTER 4. RELEVANT LOGIC PROGRAMMING

where \( \theta' > \tau \theta \), where \( \tau \) is composed of elements of the form \( t/x' \) where \( x' \) is a renaming of \( x \) in accordance with the Tip and Chain rules. Clearly \( \tau \theta \) has the correct form, and a fortiori \( \theta' \) must also, it being a trivial matter to show that for any \( u/y \in \theta', t/y \in \tau \theta \) for some \( t \).

**Ad** (\( W \vdash \)) By the Contraction and Deletion properties.

**Ad** (\( t \vdash \)) By the \( t \)-Weakening property.

**Theorem 95 (Completeness)** Let \( S \) be an assertion and let \( A \) be an unquantified query. If there is a \( \mathcal{G}L^R_{AQ} \) proof of \( \forall S \vdash \exists A \) then the set \( \Theta \) of substitutions \( \theta \) for which \( \forall S \vdash A \theta \) is provable is non-empty, and for each such \( \theta \) there is a \( \theta' \in \Theta \) such that

- \( \text{Proof}(\langle \forall S \rangle, A^\gamma, D, \{1\}, \epsilon, \theta') \) (where \( D \) is either \( \{1\} \) or \( \emptyset \) depending on whether or not \( t \in S \))

- \( A \theta = A \theta' \sigma \) for some \( \sigma \) (so \( A \theta' \) is more general than \( A \theta \)).

**Proof** The existence of \( \Theta \) follows by Th. 78. By Th. 94, for each \( \theta \in \Theta \) we have

\[
\text{Proof}(\langle \forall S \rangle, A^\gamma, D, \{1\}, \epsilon, \rho)
\]

where \( \rho \) is composed of elements of the form \( t/x \) where \( x \) is a renaming in accordance with the Tip and Chain rules. Since \( \forall S^\gamma = (\forall S^\gamma) \theta \), by \( \theta \)-Shift we know

\[
\text{Proof}(\langle \forall S \rangle, A^\gamma, D, \{1\}, \theta, \theta \rho)
\]

and by \( \theta \)-Strengthening,

\[
\text{Proof}(\langle \forall S \rangle, A^\gamma, D, \{1\}, \epsilon, \theta')
\]

for some \( \theta' > \theta \rho \). That \( \theta' \in \Theta \) follows from the Soundness theorem. Let \( \sigma = \theta \rho \). We know that \( A \theta' \sigma = A \sigma \). Further, \( A \sigma = A \theta \), since clearly none of the renaming variables substituted for by \( \rho \) can occur in \( A \theta \).
4.6 Implementation

The relevant extension to (positive) PROLOG presented here in its CQEP guise has been implemented as part of the experimental CLOGPROG system, using a simple extension to the simple stack-frame, structure sharing PROLOG interpreter as described in [Hog84].

At the user-level, assertions are of the form $A'$, where $A$ is an unquantified NLeft wff (that is, a QN-normal Left wff with no leading quantifiers), and $A'$ results from $A$ by replacing all occurrences of $\&$ and $\circ$ by $\land$, and all occurrences of $+$ and $\lor$ by a single disjunction sign (in fact we use the PROLOG-standard $'$). The operations performed by the parser are implicitly as follows:

- Convert $A'$ to $A$.
- Convert $A$ to the unquantified Left wff $B = A^{DL}$ by moving negation signs in until they occur only in atomic contexts.
- Convert $B$ to the unquantified assertion $B_1 \& \ldots \& B_m = B^A$.
- Add each elementary assertion $B_i$ to the database.

User-level queries are of the form $A'$, where $A$ is an unquantified NRight wff, $A'$ results from $A$ again by replacing all occurrences of $\&$ and $\circ$ by $\land$, and all occurrences of $+$ and $\lor$ by $'$. The parsing operations are as follows:

- Convert $A'$ to $A$.
- Convert $A$ to the unquantified Right wff $B = A^{DR}$.
- Convert $B$ to the unquantified query $B_1 \circ \ldots \circ B_m = B^Q$, and form the existential closure.

For efficiency reasons, the current implementation does not allow $t$ to appear in assertion positions. Explicit internal universal quantifiers are allowed, as defined in the syntax for QN-normal formulas.
CHAPTER 4. RELEVANT LOGIC PROGRAMMING

A switch RELEVANT/NORELEVANT controls whether implications are to be interpreted relevantly or intuitionistically, by enabling or disabling the use checks in CQEP. As pointed out above, disabling these gives a system essentially equivalent to the NPROLOG of [GR84]. Setting NORELEVANT is logically equivalent to adding $t$ to every new database segment formed in the course of the computation.

A switch REGULAR/NOREGULAR controls whether the database is to be taken as a regular or a non-regular $LR_+$ theory. Setting this switch is logically equivalent to adding $t$ to the database, as discussed above; queries which are theorems of $LR_+$ under the appropriate translation will succeed from the ‘empty’ database with RELEVANT set, but nothing will succeed if it is not. See the ‘Speculations’ section below for applications where it might be useful to have this option.

The details of the extensions to the standard interpreter model of [Hog84] may be read more or less directly from the CQEP algorithm. Instead of a single database we have a sequence of database segments, corresponding to $S_1 \ldots S_m$. The interpreter maintains a global mask $\text{Used}$ on this sequence, recording which of the segments have been used in the computation so far, and a pointer $\text{curr-seg}$ to the segment from which the current procedure was called. Every time a procedure is called the segment from which it came is masked in $\text{Used}$. When a procedure $A \rightarrow B$ is called, $A$ is added to the list of database segments and $B$ is called. At the conclusion of the subcomputation generated by this call a check is made that $A$ has been masked (corresponding to the reduction of a check-point on the goal stack). If this succeeds, $A$ is removed from the list of segments, and its mask from $\text{Used}$; otherwise backtracking occurs. Choice-point frames save the current state of $\text{Used}$ and $\text{curr-seg}$, to be restored on backtracking. If REGULAR is set we initialize $\text{Used}$ to mask the base-level database.

As input formulas are compiled the quantificational status of variables in each subformula is recorded. When a procedure is called, new space is pushed onto the stack for each variable in it, but the slots for those variables which are not marked as universally quantified are filled with a link back to a previously established environment. These environments are maintained in structures called subproofs:
when a call to $A \rightarrow B$ is made, a new subproof is generated, and space is pushed onto the stack for each variable occurring in the formula. The subproof records the stack address of this environment. If a procedure is subsequently called from the segment consisting of $A$, its unquantified variables are represented by pointers to the slots for those variables in the environment recorded in the subproof, maintaining the linkage between variables in antecedents and consequents. Quantified variables in the called procedure are allocated new space. This corresponds to the selective renaming of variables in the Rename procedure.

One of the goals of the implementation is to make the overheads caused by the extension invisible for normal PROLOG use. The basic engine without extensions runs at about 1400 lips on a Sun-3 (by the naive reversal test); with the full CLOGPROG system this is reduced to about 1300 lips.

The benefits of restricting the fragment of $LR_+$ in the computation may be illustrated by comparing the performance of CLOGPROG and Kripke (the full theorem prover for propositional $LR$) on twenty randomly generated theorems in the L/R restricted vocabulary. Kripke needed over 300 seconds CPU time to prove all twenty, whereas CLOGPROG took less than 0.3 seconds. Although the value of such a simple comparison is somewhat dubious, we take it that these results do much to justify the decrease in expressive power caused by the L/R restrictions.

4.7 Speculations

4.7.1 Loop Detection

Here we flesh out the loop-detection example of Chapter 1. Take a database without $t$ (ie, with NOREGULAR set in CLOGPROG) and ask the question $A \rightarrow A$. If this succeeds, we know that there is at least a chance that the query $A$ will loop, because the success of $A \rightarrow A$ means that there is a computation path for $A$ which generates another call to $A$. Note that the success of $A \rightarrow A$ from a REGULAR database will always succeed (and so tells us nothing about whether
or not \( A \) loops), since there is then no requirement that any elements of the database (except the implicit \( t \)) be used.

This of course is not a complete loop detection strategy, nor even a particularly useful one as it stands. The query \( f(x) \mapsto f(x) \) will fail to terminate for any recursive \( f \) (if our system implements an occurs-check). For example, consider the standard ‘append’ predicate:

\[
\text{append}([], x, x)
\]

\[
\text{append}(x, y, z) \rightarrow \text{append}([w|x], y, [w|z])
\]

The call \( \text{append}(t, u, v) \rightarrow \text{append}(t, u, v) \) will result in the following computation:

1 Form a new segment with \( \text{append}(a, b, c) \), compute \( \text{append}(a, b, c) \).

2 Initial attempts to match the goal with the fact clause in the base segment and with the clause in the added segment result in failed use checks, so we backtrack and match the goal with the rule clause, renamed as

\[
\text{append}(x_1, y_1, z_1) \rightarrow \text{append}([w_1|x_1], y_1, [w_1|z_1])
\]

to generate the substitution \( \{[w_1|x_1]/t, y_1/u, [w_1|z_1]/v\} \).

3 The new goal is \( \text{append}(x_1, y_1, z_1) \) and the clause in the added segment with the substitution applied is \( \text{append}([w_1|x_1], y_1, [w_1|z_1]) \); the occurs-check prevents these from unifying, and clearly the computation will never terminate.

However, if the query is instead \( \text{append}(t_1, u_1, v_1) \rightarrow \text{append}(t_2, u_2, v_2) \) (variables uninstantiated) the occurs-check does not intrude and the (first) answer will be \( t_1 = x_1, u_1 = y_1, v_1 = z_1, t_2 = [w_1|x_1], u_2 = y_1, v_2 = [w_1|z_1] \). If we examine the bindings found for \( t_1, u_1, v_1 \) we can interpret this answer as saying that a call \( \text{append}(t, u, v) \) will generate a call to \( \text{append}(t', u', v') \), where \( t', u', v' \) are no simpler (in an intuitive sense) than \( t, u, v \), and so there is a danger of looping. Of
course the ordering on clauses combined with the fixed selection scheme means that the computation does not actually loop (since the fact clause in the base segment will always be selected first), but if we read the program purely declaratively there is an infinite computation path, and this is the information returned by the query.

Now consider the same query except with \( t_2 \) instantiated to \([a, b]\), say. The values returned for the antecedent variables are \( t_1 = [b], u_1 = y_1, v_1 = z_1 \) indicating that a call to \texttt{append}([a, b], u, v) generates a simpler call to \texttt{append} and we would not expect there to be an infinite computation path (as indeed there is not). Similarly for a call \texttt{append}(t, u, [a, b]), but the implicational query with \( u_2 \) instantiated to \([a, b]\) returns antecedent bindings \( t_1 = x_1, u_1 = [a, b], v_1 = z_1 \) indicating that the call \texttt{append}(t, [a, b], v) has an infinite computation.

Of course this rather vague discussion does not by itself establish the usefulness of the relevant conditional as a loop detection device, but we think the prospects are interesting.

### 4.7.2 Negation as Failure

There is in principle no barrier to adding negation as failure (NAF) to our system. Clearly we could add enough of NAF to the system to make it behave like normal (full) PROLOG for normal PROLOG operations; the conceptual difficulty lies in how to interpret queries like \( A \rightarrow \neg B \) (where we let \( \neg \) stand for NAF): ‘If \( A \) is true then \( B \) therefore fails’, but the precise meaning of such a statement still eludes us. We think that NAF is one of logic programming’s most interesting contributions to logic, adding a kind of ‘constructive non-demonstrability’ to Curry’s excellent discussion of negation in [Cur77, Ch.6]. The extension of this notion beyond the first-degree level of PROLOG is a promising avenue for future work.
Chapter 5

Counterfactuals

5.1 Introduction

In this chapter we will extend our treatment to the counterfactual conditional, for which we will use the symbol ‘\(\rightarrow\)’. We will only allow counterfactuals in query positions; implications in assertion positions will always be indicative. Recall from Chapter 1 the Ramsey Test:

\[ \alpha \vdash A \rightarrow B \iff \alpha + A \vdash B \]

The counterfactual differs from the indicative in that the ‘+’ operation must include some mechanism for removing from \(\alpha\) those elements which are uncotenable, in some sense, with \(A\). A solution to the problem of counterfactuals in general relies on the precise definition of ‘cotenability’ in this context. Further, since there will usually be more than one ‘minimal mutilation’ of \(\alpha\) which leaves a cotenable residue, we have the problem of selecting among these choices.

5.2 Syntax

We extend the definitions of elementary queries of Sec. 4.2.1 as follows:
• If $A$ is an assertion and $B$ is an unquantified query then $A \rightarrow B$ is an elementary query.

So $A \rightarrow B$ is an elementary query whenever $A \rightarrow B$ is.

From Sec. 4.3.5 we know that whenever $B$ is an elementary assertion, $\neg B$ is a (QN-normal) NRight wff. Throughout the development, where $B$ is an elementary assertion we will use $\neg B'$ as shorthand for $(\neg B)^{DR}$ (see Def. 20). Specifically:

• If $B$ is an atom $p$ then $\neg B'$ is $q$, if $p = \neg q$ for some $q$, and $\neg p$ otherwise.

• If $B$ is an implication $C \rightarrow p$ (where $p$ is an atom), then $\neg B'$ is $C \circ q$, if $p = \neg q$ for some $q$, and $C \circ \neg p$ otherwise.

We know that for any elementary assertion $B = C \rightarrow p$, $C$ is an unquantified query and $p$ is an atom. So $(\neg B)^{DR}$ (as given here) will be an unquantified query; either an atom, or of the form $C \circ q$ where $q$ is an atom.

In order to focus on the special problems of counterfactuals, for most of this chapter we will restrict the discussion to the propositional language. The major additional apparatus necessary to extend the development to the first-order level is some kind of exception apparatus, similar to those proposed for sound forms of negation as failure (see [Kun87] and Sec. 5.8 for more details).

Where $T$ is an (unquantified assertion), in other words a conjunction $A_1 \land \ldots \land A_m$ of elementary elements, we will often treat $T$ as the set $\{A_1, \ldots, A_m\}$. For example, we might say $A_i \in T$ when we really mean $A_i$ appears as a conjunct in $T$, or speak of an object $T - \{A_i\}$ where what is intended is the result of deleting all occurrences of $A_i$ from $T$. The mapping between the set-theoretic locutions and their formula counterparts is straightforward.

For convenience, we will often use $A_1 \circ A_2 \circ \ldots \circ A_n \rightarrow B'$ as shorthand for queries of the form $A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_n \rightarrow B'$; of course the two forms are equivalent with respect to the underlying logic, $LR_+$. 


5.3 Cotenability

From our statement of the Ramsey Test we can extract the following:

**Definition 23** $\alpha \vdash A \rightarrow B$ iff $\text{Cot}(A, \alpha), A \vdash B$ iff $\text{Cot}(A, \alpha) \vdash A \rightarrow B$.

where by $\text{Cot}(A, \alpha)$ we mean 'the part of $\alpha$ cotenable with $A$'. So the counterfactual $A \rightarrow B$ follows from $\alpha$ iff the indicative $A \rightarrow B$ follows from the part of $\alpha$ which is cotenable with $A$.

Given an evaluation procedure for the indicative conditional a computational strategy for counterfactuals will depend on the precise definition of cotenability. The strategy we adopt owes its origins to Goodman's notion of cotenability in [Goo83]. The essence of this is as follows: Assume that $\alpha$ is the single-element multiset $[T]$ (where $T$ an unquantified assertion). Then some $C \in T$ is cotenable with $A$ just where it is not the case, that were $A$ true then $C$ would be false. Formally, $C \in \text{Cot}(A, T)$ iff $C \in T$ and $T \not\vdash A \rightarrow \neg C$. And by our statement of the Ramsey Test, this in turn becomes: $C \in \text{Cot}(A, T)$ iff $C \in T$ and $\text{Cot}(A, T) \not\vdash A \rightarrow \neg C$.

**Definition 24** An $A$-restriction of a set of elementary assertions $T$ is any set $T'$ obeying the following condition: $C \in T'$ iff $C \in T$ and $T' \not\vdash A \rightarrow \neg C$.

So $\text{Cot}(A, T)$ is some $A$-restriction of $T$. Leaving aside the question of the general existence of $A$-restrictions, in any case we can see that there will generally be more than one, for any $A$ and $T$. For example, if $T$ contains just the following elements:

- $A \land B \rightarrow \neg C$
- $A \land C \rightarrow \neg B$
- $B$
- $C$

then the $A$-restrictions are as follows:
• \( T_1 = \{ A \land B \rightarrow \neg C, A \land C \rightarrow \neg B, B \} \)

• \( T_2 = \{ A \land B \rightarrow \neg C, A \land C \rightarrow \neg B, C \} \)

• \( T_3 = \{ B, C \} \)

as readers may verify for themselves.

**Definition 25** Where \( A \) is an assertion and \( T \) is a set of elementary assertions, \( \text{REST}(A, T) \) is the set of \( A \)-restrictions of \( T \).

So far we have ignored the question of what \( \text{Cot}(A, \alpha) \) means when \( \alpha \) contains more than one element. Until Sec. 5.7, we will restrict the discussion to the case where every proof has at most two elements on the left of the \( \rightarrow \) in every sequent, or in computational terms, to the case where the database never contains more than two segments. So the only \( \text{Cot}(A, \alpha) \) objects are those which have \( \alpha \) as a single element multiset \([T]\), where \( T \) is a set of elementary assertions.

We can use \( \text{REST}(A, T) \) in two different ways: either we can stick with our formulation of the Ramsey Test given above and stipulate that \( \text{Cot}(A, T) \) is to be some most preferred element of \( \text{REST}(A, T) \), or we can modify the statement as follows:

**Definition 26** \( T \vdash A \rightarrow B \) iff \( T' \vdash A \rightarrow B \) for every \( T' \) which is an element of some preferred subclass of \( \text{REST}(A, T) \).

This revised form seems intuitively preferrable; in fact, our computation will be (at best) incomplete with respect to both forms, but for the moment the important aspect is that they both involve making some value judgements with respect to \( \text{REST}(A, T) \), to find some subclass of 'best' elements, either a singleton or a set of elements which are each as good as each.

Before going on to discuss rankings, we should point out the role of relevance in our definition of \( A \)-restrictions. For an irrelevant \( \rightarrow \), \( T' \in \text{REST}(A, T) \) only if \( T' \cup \{ A \} \not\vdash \neg B \) for all \( B \in T' \); however this is not true for a relevant \( \rightarrow \). Consider a \( T \) containing the following information:
CHAPTER 5. COUNTERFACTUALS

p1 John is Fred's father.

p2 John is not Fred's father.

p3 John did not pass maths.

p4 If John studied hard then John passed maths.

Let \( A \) be 'John studied hard'. For a relevant \( \rightarrow \), the elements of \( \text{REST}(A, T) \) are \{p1, p2, p3\} and \{p1, p2, p4\}. The inconsistency between p1 and p2 is unrelated to \( A \) and can be ignored. An irrelevant reading, on the other hand, would give the following \( \text{REST}(A, T) \): \{p1, p3\}, \{p2, p3\}, \{p1, p4\}, \{p2, p4\}. Either p1 or p2 has to go. It seems to us that the relevant reading has a clear conceptual advantage; the fact that p1 and p2 are inconsistent has nothing to do with \( A \) and should have no bearing on the question of the subsets of \( T \) cotenable with \( A \). The relevant reading focuses attention on the part of the database which is material to the issue at hand. To push the idea, it seems sensible to work with the relevant version of cotenability even if we want our counterfactual to be irrelevant. (Note that even though the elements of \( \text{REST}(A, T) \) with \( \rightarrow \) taken relevantly are inconsistent with respect to '¬' for the example given, this is not to say that everything follows from them in a classical or intuitionistic setting, since our '¬' is not the standard connective of these systems but rather an inconsistency tolerating negation, which can be added to the positive fragments of these logics in a conservative way.) For the rest of this paper, we will only be concerned with relevant cotenability.

5.4 Outline of the Strategy

In this section we present a strategy for selecting a preferred subclass from \( \text{REST}(A, T) \). The main goal is to find a scheme which will allow a reasonably clean and efficient computation while remaining plausible and coherent with respect to some simple intuitions. We claim no philosophical virtue for our formulation; indeed, one of our aims has been to reduce philosophical commitment to a minimum. Value judgements are introduced in an entirely operational way;
specifically, value is represented in our system by an ordering ‘□’ on elementary assertions, where \( A □ B \) is to be interpreted as: ‘In a contest between \( A \) and \( B \), \( A \) wins’. There is no commitment to any particular theory of value (e.g., plausibility as in [Res76], similarity as in [Lew73] and [Sta68] or preference as in [vW63]) which would tell us why \( A \) should win the contest. Contests themselves are defined proof-theoretically; for \( B, C ∈ T \) there is a contest between \( B \) and \( C \) (in the context of \( A \)) just in case \( T ⊢ A ∧ B → ¬ C \) and \( T ⊢ A ∧ C → ¬ B \). So we are not committed to any interpretation which would require certain properties of ‘□’. In short, our approach to counterfactuals is driven by the pragmatic desire to find ways of computing them based on some simple operational intuitions. We do not claim to have any answers to the larger (and more important) philosophical problems of counterfactuals.

The broad details of the strategy are as follows. We start with a set \( T \) of elementary assertions, and the goal is to find those elements of \( T \) which are compatible with some assertion \( A \). We use our notion of contests and the ‘□’ ordering to divide \( T \) up into ‘winners’ and ‘losers’. Generally the ‘winners’ and ‘losers’ will not partition \( T \); there will remain some ‘doubtful’ elements. We then consider the class of those \( T' ∈ \text{REST}(A, T) \) which contain all of the ‘winners’ and none of the ‘losers’. This, if we like, can be the preferred subclass of \( \text{REST}(A, T) \) which we use for evaluating counterfactuals; on the other hand, there is no reason why we could not seek to further reduce the set by applying other value considerations. So we do not think of our strategy as necessarily complete; it may be just a way of restricting the choices for the preferred subclass. In the extreme case where the ‘winners’ and ‘losers’ do partition \( T \), all these approaches coincide; we have a single preferred \( A \)-restriction, namely the the set of ‘winners’, and no further ranking is necessary.

### 5.5 Contest Divisions

Let \( T \) be a set of elementary assertions. The goal is to find a pair \((\text{Win}, \text{Lose})\) where \( \text{Win} \) and \( \text{Lose} \) are respectively the ‘winners’ and the ‘losers’ (both subsets
of \( T \).

The first condition on these sets is:

\[ \text{Cont1 } \text{Win} \cap \text{Lose} = \emptyset. \]

Naturally, nothing is both a winner and a loser. The winners include all of those elements of \( T \) which are not refuted by a good argument:

\[ \text{Cont2 } \text{Win} = \{ C : C \in T, T - \text{Lose} \not\models A \rightarrow \neg C \} \]

A good argument is one which does not use any elements which have been assigned the status of losers. Dually, the losers in \( T \) include those elements which are refuted by an unbeatable argument:

\[ \text{Cont3 } \text{Lose} = \{ C : C \in T, \text{Win} \not\models A \rightarrow \neg C \} \]

An unbeatable argument is one which uses only elements which are assigned the status of winners. The conditions Cont1 to Cont3 do not uniquely define winners and losers. Consider the previous example where \( T \) consisted of:

\begin{align*}
\text{p1 } & A \land B \rightarrow \neg C \\
\text{p2 } & A \land C \rightarrow \neg B \\
\text{p3 } & B \\
\text{p4 } & C
\end{align*}

Then the pairs which satisfy Cont1 to Cont3 are as follows:

- \( \text{Win} = \{ p1, p2, p3 \}, \text{Lose} = \{ p4 \} \)
- \( \text{Win} = \{ p1, p2, p4 \}, \text{Lose} = \{ p3 \} \)
- \( \text{Win} = \{ p3, p4 \}, \text{Lose} = \{ p1, p2 \} \)
- \( \text{Win} = \text{Lose} = \emptyset \)
CHAPTER 5. COUNTERFACTUALS

as readers may verify for themselves.

Clearly the machinery will need some more ‘muscle’ if it is to be useful for anything. To this end, we introduce a relation on elementary assertions, which we write as ‘$A\sqsupset B$’ and read as ‘In a contest between $A$ and $B$, $A$ wins.’ The only condition which we place on $\sqsupset$ is that it be asymmetric (ie, for all $A$, $B$, it is never the case that $A\sqsupset B$ and $B\sqsupset A$); we can, if we like, induce $\sqsupset$ from a non-asymmetric $\sqsubseteq$ by the definition: $A\sqsupset B$ iff $A\sqsupseteq B$ and $B \not\sqsubseteq A$. Note that this relation carries no philosophical baggage; we are not interested in why $A$ should win the contest, merely that it does.

**Definition 27 (Contests)** Where $T$ is a set of elementary assertions and $B, C \in T$ then there is a contest between $B$ and $C$ (in the context of $A$ just in case $T \vdash A \land B \rightarrow \neg C$ and $T \vdash A \land C \rightarrow \neg B$.

Since ‘$\land$’ in these positions is in fact fusion, we can write this in perhaps more graphic form: a contest exists between $B$ and $C$ just in case $T \vdash B \rightarrow (A \rightarrow \neg C)$ and $T \vdash C \rightarrow (A \rightarrow \neg B)$. Intuitively, $B$ forces $C$ to be incompatible with $A$, and vice versa. In the above example, there are contests between each of $p_1$ and $p_2$ on the one hand, and each of $p_3$ and $p_4$ on the other; also between $p_3$ and $p_4$.

We want to find plausible conditions which will extend Cont1 to Cont3 in a way which uses the notion of a contest, and the $\sqsupset$ ordering. If in the example we had $p_1 \sqsupset p_3$, $p_1 \sqsupset p_4$, $p_2 \sqsupset p_3$ and $p_2 \sqsupset p_4$ then the intent would be clear. Both $p_1$ and $p_2$ win all of their contests, so intuitively we would expect either the choice for ‘the’ contest division of $T$ to be between the first and second of the listed pairs; further we would expect to select the first by setting $p_3 \sqsupset p_4$, and the second by setting $p_4 \sqsupset p_3$. So in this case adopting a simple strategy, in which elements which win all of their contests are winners and those which lose all of theirs are losers, would work. But we should not expect things to be so straightforward in the general case.

**Example 1**

Let $T$ contain the following:
The contest here are between p5 and each of p1, p2 and p6; between p6 and each of p1 to p7; and between p7 and each of p3, p4 and p6. Set the ordering so that each implication, p1 to p4, wins every contest; on the suggested strategy, they all becomes winners. Now set \( p5 \subseteq p6 \) and \( p6 \subseteq p7 \). On the suggested strategy, p7 loses every contest and so becomes a loser; but by Cont2, if p7 is a loser then p6 must be a winner (because \( T - \{p6\} \not\models A \rightarrow \neg D \)), which by Cont3 is enough to make p5 a loser as well. But this would not seem to be intuitively correct; p7 is beaten by p6, but p6 is in turn beaten by p5. So there is a clear sense in which p7's loss to p6 is not 'essential', but somehow 'masked' by p5's victory. We take it as intuitively correct in this situation for p5 to win, and hence for p6 to lose (by Cont3), and finally therefore for p7 to win (by Cont2).

The formalization of this idea of this intuition of 'essential' and 'inessential' contests requires a slightly more sophisticated approach. To this end, let us consider all the sets \( X \) of the form \( \{B_1 \ldots B_m\} \subseteq T \; \text{s.t.} \; \vdash A \circ B_1 \circ \ldots \circ B_m \rightarrow \neg C \). We will call such a set a \textit{refutation} of \( C \) (in the context of \( A \)). If any refutation is a subset of \( \text{Win} \) then \( C \in \text{Lose}_1 \), by Cont3. If \( X \cap \text{Lose} \neq \emptyset \) for all refutations \( X \) then \( C \in \text{Win} \), by Cont2. So let us consider just those refutations which consist only of elements which do \textit{not} lose, \textit{except possibly if C wins}. It is certainly a necessary requirement that some such refutation must exist, if \( C \) is not to win. However if each such refutation contains some \( B_i \) for which \( \text{Win} \vdash A \circ C \rightarrow B_i \) (and which would therefore lose if \( C \) were to win), and furthermore \( C \) beats \( B_i \), this would
CHAPTER 5. COUNTERFACTUALS

seem to be a minimal sufficient condition for $C$ to be assigned the status of a winner. Every possible argument against $C$ uses some element which $C$ beats in a contest, using (apart from $C$ itself) only elements which are already taken to win.

Our new condition for winners then becomes:

**Cont4 Win** is the set of $C \in T$ which satisfy the following condition: For all \( \{B_1 \ldots B_m\} \subseteq T \) s.t.

- $A \circ B_1 \circ \ldots \circ B_m \rightarrow \neg C$
- $\text{Win} - \{C\} \not\models A\rightarrow \neg B_i$ for all $B_i$

there is some $B_i$ for which

- $\text{Win} \vdash A\circ C \rightarrow \neg B_i$
- $C \sqsubseteq B_i$

For every such $B_i$, there is of course a contest between $B_i$ and $C$. For if $B_i$ is an element of a possible refutation for $C$ then $T \vdash A \circ B_i \rightarrow \neg C$, and if $\text{Win} \vdash A \circ C \rightarrow \neg B_i$ then of course $T \vdash A \circ C \rightarrow \neg B_i$. This condition then defines a class of ‘critical contests’ for $C$; winning these contests is a sufficient condition for $C$ to be assigned the status of a winner.

The new condition for losers follows by similar considerations. Again we consider refutations $X$ of $C$ in $T$. If $C$ is to lose, then there must be some $X$ which consists only of elements which do not fail to win except possibly if $C$ does not lose. Consider some such $X$. If every $B_i \in X$ for which $T - \text{Lose} \vdash A \circ C \rightarrow \neg B_i$ (that is, those $B_i$ which do in fact fail to win if $C$ does not lose) beats $C$, then it seems we have a minimal sufficient condition for $C$ to be classed amongst the losers. For we have a refutation of $C$ which uses only elements $B_i$ which are either already taken as winners, or else can only be refuted by using $C$, and $C$ loses its contest with any such $B_i$.

**Cont5 Lose** is the set of $C \in T$ satisfying the following: For some \( \{B_1 \ldots B_m\} \subseteq T \) s.t.
CHAPTER 5. COUNTERFACTUALS

- \( \vdash A \circ B_1 \circ \ldots \circ B_m \rightarrow \neg C \)
- \( T - (\text{Lose} \cup \{C\}) \not\vdash A \rightarrow \neg B_i \) (for each \( B_i \))

for each \( B_i \), if \( T - \text{Lose} \vdash A \circ C \rightarrow \neg B_i \) then \( B_i \sqsubseteq C \).

Again, for every appropriate \( B_i \) there is of course a contest between \( C \) and \( B_i \), and so this condition serves to define a class of 'critical losing contests': losing these is sufficient for \( C \) to be classed as a loser.

Cont4 and Cont5 seem to be minimal conditions for winners and losers, given our interpretation of the '\( \sqsubseteq \) ' ordering.

Our final condition is:

**Cont6** (Win, Lose) is the smallest pair satisfying Cont1 to Cont5.

We will call the pair satisfying Cont1 to Cont6 the contest division of \( T \) (in the context of \( A \) and \( \sqsubseteq \)).

The following construction demonstrates the existence of the contest division for all \( T, A \) and \( \sqsubseteq \):

- Let \( \text{Win}^0 = \text{Lose}^0 = \emptyset \). Let \( \text{Doubt}^0 = T \). At each succeeding stage of the construction, 'doubtful' elements are distributed amongst the winning and losing sets until a stable point is reached.

- \( \text{Win}^{n+1} \) is the union of \( \text{Win}^n \) with the set of all \( C \in \text{Doubt}^n \) satisfying the following: For all \( \{B_1 \ldots B_m\} \subseteq \text{Doubt}^n \) s.t. \( \text{Win}^n \vdash A \circ B_1 \circ \ldots \circ B_m \rightarrow \neg C \), there is some \( B_i \) s.t. \( \text{Win}^n \vdash A \circ C \rightarrow \neg B_i \) and \( C \sqsubseteq B_i \).

- \( \text{Lose}^{n+1} \) is the union of \( \text{Lose}^n \) with the set of all \( C \in \text{Doubt}^n \) satisfying the following: For some \( \{B_1 \ldots B_m\} \subseteq \text{Doubt}^n \) s.t. \( \text{Win}^n \vdash A \circ B_1 \circ \ldots \circ B_m \rightarrow \neg C \), for each \( B_i \), \( T - (\text{Lose}^n \cup \{C\}) \not\vdash A \rightarrow \neg B_i \) and \( B_i \sqsubseteq C \).

- \( \text{Doubt}^{n+1} = \text{Doubt}^n - (\text{Win}^{n+1} \cup \text{Lose}^{n+1}) \).

Clearly this construction must reach a stable point for some finite \( k \) at which no further distribution of 'doubtful' elements is possible (given a finite \( T \)). The
proof that \((\text{Win}^k, \text{Lose}^k)\) is the contest division uses the following facts, which are easily verified:

- For each stage \(n\), \(T = \text{Win}^n \cup \text{Lose}^n \cup \text{Doubt}^n\), and \(\text{Doubt}^n\) is disjoint from both \(\text{Win}^n\) and \(\text{Lose}^n\).

- If \(C \in \text{Win}^n\) then \(T - \text{Lose}^{n+1} \not\models A \rightarrow \neg C\).

- If \(C \in \text{Lose}^n\) then \(\text{Win}^{n+1} \models A \rightarrow \neg C\).

The proof itself proceeds as follows:

**Ad Cont1** At the 0-th stage the winning and losing sets are both empty and so trivially disjoint. An element \(C\) of \(\text{Doubt}^n\) will never be given to both \(\text{Win}^{n+1}\) and \(\text{Lose}^{n+1}\) at the \(n+1\)-st stage of the construction; if it were then it would have to be the case that \(C \subseteq B_i\) and \(B_i \subseteq C\) for one of the indicated \(B_i\), but this is impossible, since \(\subseteq\) is asymmetric. So clearly if the the winners and losers at the \(n\)-th stage are disjoint, the same will be true at the \(n+1\)-st stage, and this completes the induction.

**Ad Cont2** If \(C \in \text{Win}^k\) then by one of the facts noted above, \(T - \text{Lose}^{k+1} \not\models A \rightarrow \neg C\), and since \(k\) represents the stable state of the construction, \(T - \text{Lose}^k \not\models A \rightarrow \neg C\). For the other direction, let \(C \in T\) and assume \(T - \text{Lose}^k \not\models A \rightarrow C\). We will show that \(C \not\in \text{Doubt}^k\) and \(C \not\in \text{Lose}^k\) leaving \(C \in \text{Win}^k\) as the only possibility.

- For reductio, assume that \(C \in \text{Doubt}^k\). Since winners and losers are disjoint, it must be the case that \(\text{Win}^k \subseteq T - \text{Lose}^k\); hence by hypothesis there are no \(B \in \text{Doubt}^k\) for which \(\text{Win}^k \models A \circ B \rightarrow \neg C\), and so by the construction it must be the case that \(C \in \text{Win}^{k+1}\). Since the \(k\)-th stage represents the fixed point of the construction, this means that \(C \in \text{Win}^k\), contradicting the reductio assumption.

- For reductio, assume that \(C \in \text{Lose}^k\). \(C\) must have been assigned to the losers at some stage \(n + 1 \leq k\) of the construction. By the construction, there is a \(\{B_1 \ldots B_m\} \subseteq \text{Doubt}^n\) s.t. \(T - \text{Lose}^n \models \)}
AoB_1 o ... oB_m → ¬C and T → (Lose^n ∪ {C}) ∤ A→¬B_i for each B_i. But then at the n + 2-nd stage there will be no D ∈ Doubt^{n+1} for which Win^{n+1} ⊨ AoD→¬B_i for any of these B_i; hence each B_i is an element of Win^{n+2} and so of T - Lose^k, and straightforwardly we can conclude that T - Lose^k ∨ A→¬C, contradicting the hypothesis.

**Ad Cont3** Using similar methods to those employed in the previous case.

**Ad Cont4**

- Show that every C ∈ Win^k satisfies the conditions of Cont4. Assume that C ∈ Win^k and that there is a \{B_1 ... B_m\} ⊆ T s.t. ⊨ AoB_1 o ... oB_m→¬C and Win^k - {C} ∤ A→¬B_i for all B_i. Clearly it is the case that Win^k ⊨ AoC→¬B_i for some B_i; otherwise Win^k ∤ A→¬B_i for each B_i, and so T - Lose^k ⊨ A→¬C, which is impossible if C ∈ Win^k. For reductio, assume that C ∤ B_i for each such B_i.

Now C must have been assigned to the winners at some stage n + 1 of the construction, and we have \{B_1 ... B_m\} ⊆ Doubt^n and for some B_j, Win^n ⊨ AoC→¬B_j and C ⊆ B_j. But if Win^n ⊨ AoC→¬B_j then of course Win^k ⊨ AoC→¬B_j, and this contradicts the reduction assumption.

- Show that every C which satisfies the conditions of Cont4 is an element of Win^k. Assume that all the conditions of Cont4 are met for C ∈ T by Win^k. Again we must show that C ∉ Doubt^k and C ∉ Lose^k.

  - Assume for reductio that C ∈ Doubt^k. First note that for all B ∈ Doubt^k it is not the case that Win^k - {C} ⊨ A→¬B; otherwise we would have B ∈ Lose^{k+1} which contradicts the assumption that the k-th stage represents a fixed point. Similarly it is not the case that Win^k ⊨ AoC→¬B for any B ∈ Win^k, for the construction guarantees that any refutation of such a B must use some losing element. Now let \{B_1 ... B_m\} ⊆ Doubt^k and Win^k ⊨ AoB_1 o ... oB_m→¬C. Under the conditions of the hypothesis it is then clear that for some B_i, Win ⊨ AoC→¬B_i and C ⊆ B_i. By the construction, C ∈ Win^{k+1}, contradicting the reductio assumption.
Assume for reductio that $C \in \text{Lose}^k$. Then $C$ was assigned a losing status at some stage $n + 1 \leq k$, at which point there was a $\{B_1 \ldots B_m\} \subseteq \text{Doubt}^n$ such that $\text{Win} \vdash A \circ B_1 \circ \ldots \circ B_m \rightarrow \neg C$ and for each $B_i$, $T - (\text{Lose}^n \cup \{C\}) \not\vdash A \rightarrow \neg B_i$ and $B_i \subseteq C$. Clearly $\text{Win}^k - \{C\} \not\vdash A \rightarrow \neg B_i$ for each $B_i$, and the hypothesis tells us (with some simple moves) that for some $B_i$, $C \subseteq B_i$. But we also have $B_i \subseteq C$, which is impossible, and so we have contradicted the reductio assumption.

**Ad Cont5** Using similar methods to those employed in the previous case.

**Ad Cont6** The winning and losing sets at the 0-th stage are empty and so will be contained in their counterparts in any quadruple obeying Cont1 to Cont5. If we assume that this also the case for all stages up to $n$, then it is easy to show that any element added at the $n + 1$-st stage must be respectively a winner or a loser under the conditions of Cont4 and Cont5.

To illustrate that the machinery performs as advertised, consider Example 1 above. Recall that the intuitively correct contest division had $\text{Win} = T - \{p6\}$ and $\text{Lose} = \{p6\}$. It is simple to see that $p6 \in \text{Lose}^1$: for we have $\vdash A \circ p1 \circ p5 \rightarrow \neg p6$, $T - \{p6\} \not\vdash A \rightarrow \neg p1$ and $T - \{p6\} \not\vdash A \rightarrow \neg p5$, and also $p1 \subseteq p6$ and $p5 \subseteq p6$. And then all other elements are assigned to $\text{Win}^2$, since every refutation of each of these uses $p6$.

**Theorem 96** If the contest division partitions $T$ then

- $\text{Win} = \{C : C \in T \& \text{Win} \not\vdash A \rightarrow \neg C\}$

- $\text{Lose} = \{C : C \in T \& T - \text{Lose} \vdash A \rightarrow \neg C\}$

**Proof** Immediate from Cont2 and Cont3.

So in the extreme case where everything in $T$ either wins or loses, the contest division will deliver a unique preferred element of $\text{REST}(A, T)$, namely $\text{Win}$ itself. Unfortunately, this is not a state of affairs which can always be attained. Consider a set $T$ consisting of the following:
Assume that the assertions are related as follows: $p_1 \sqsubseteq p_5$, $p_2 \sqsubseteq p_6$, $p_3 \sqsubseteq p_4$. Readers may satisfy themselves that there is no extension to this ordering which will deliver a contest division which partitions $T$.

For a given $\sqsubseteq$, $A$ and $T$ let $(\text{Win}, \text{Lose})$ be the contest division. Then $\text{WIN}(A, T)$ is the set of $T' \in \text{REST}(A, T)$ satisfying the following conditions:

- $\text{Win} \subseteq T'$
- $T' \cap \text{Lose} = \emptyset$

(In fact each of these conditions implies the other.) Our preferred version of the Ramsey Test is then:

$T \vdash A \rightarrow C$ iff $T', A \vdash C$ for all $T' \in \text{WIN}(A, T)$

### 5.6 The Computation

Here we present two computational models. Unfortunately we have not as yet been able to prove the equivalence of these and the theoretical development of the preceding section, and until we have the result these models are best viewed as an approach to counterfactuals following the same lines as the contest division formulation, but not necessarily identical to it.

If we implement a computational scheme which (assuming termination) answers 'Yes' to a query $A \rightarrow B$ if and only if $\text{Win} \vdash A \rightarrow B$, and 'No' if and only if
T - Lose \not\rightarrow A \rightarrow B$, then the answers are correct with respect to any preferred subclass of $\text{REST}(A, T)$ all of whose elements contain $\text{Win}$. Where neither of these conditions holds, the computation should report the existence of unresolved contests. This provides the framework for the computation; whether or not it matches it exactly is still an open question.

5.6.1 Algorithmic Description

We first give an algorithmic description of the model. The basic strategy, given a database $T$ and a counterfactual query $A \rightarrow C$, is as follows:

1 Initialize $\text{RESULT}$ to 'No'.

2 Find a proof of $A \rightarrow C$ from $T$, using the evaluation procedure described in Chapter 4. If this fails, return $\text{RESULT}$, which at this point will be either 'No' or 'Looped': either we have demonstrated that no proof of $A \rightarrow C$ from $T$ uses just elements of $T - \text{Lose}$ (for the 'No' result; this will of course be the case when there is no proof of $A \rightarrow C$ from $T$ at all), or that no proof uses just elements of $\text{Win}$, but some proof uses just elements of $T - \text{Lose}$ (for the 'Looped' result). Otherwise, proceed to step 3.

3 This step is a post-hoc check on each element $B$ of $T$ used in the proof constructed at step 1, to determine whether $B \in \text{Win}$, $B \in \text{Lose}$, or $B \in T - \text{Win} \cup \text{Lose}$. If each $B$ is an element of $\text{Win}$, we have demonstrated that $\text{Win} \vdash A \rightarrow C$ and hence $T \vdash A \rightarrow C$, and we return 'Yes' as the answer to the computation as a whole. If any $B$ is an element of $\text{Lose}$, return to step 2 and attempt to construct another proof of $A \rightarrow C$. Finally, if some $B$ is an element of neither $\text{Win}$ nor $\text{Lose}$ (and no element is an element of $\text{Lose}$), set $\text{RETURN}$ to 'Looped' and backtrack to step 2; at 'worst' the result of the whole computation will be 'Looped', since we have found a proof of $A \rightarrow C$ from $T - \text{Lose}$.

Call this procedure 'COUNT-PROOF'. The post-hoc check in step 3 is essentially a recursive application of COUNT-PROOF itself, except that we add
machinery for resolving contests. Given \( B \) to check, we put the query \( A \rightarrow \neg B \) to \( T \); if it returns ‘Yes’ we have demonstrated that \( \text{Win} \vdash A \rightarrow B \) and hence \( B \in \text{Lose} \); if it returns ‘No’ we have \( T \not\vdash A \rightarrow B \) and hence \( B \in \text{Win} \); if it returns ‘Looped’ we know that \( B \in T \in \text{Win} \cup \text{Lose} \). Call the check procedure ‘COUNT-CHECK’.

The difference between COUNT-PROOF and COUNT-CHECK comes in step 3, where we preempt the recursive call to COUNT-CHECK under certain circumstances. Each call to COUNT-CHECK has a parameter \( \kappa \), a list of elementary assertions. The first element of this list, \( \kappa[1] \), is the element to be checked, and \( \kappa[2] \) is the element checked by the calling invocation of COUNT-CHECK. Remaining elements of \( \kappa \) are the elements checked by each ancestor invocation. For the call to COUNT-CHECK in step 3 of the top-level COUNT-PROOF procedure, \( \kappa \) is set to NULL. For example, say that the top level query is \( A \rightarrow C \). We find a proof of \( A \rightarrow C \) using some element \( B \) of \( T \).

As part of step 3 of COUNT-PROOF, we call COUNT-CHECK with \( \kappa \) set to a single element list containing \( B \). This call to COUNT-CHECK involves finding proofs of \( A \rightarrow B \) from \( T \); for elements \( D \in T \) in the proof of \( A \rightarrow B \) we recursively call COUNT-CHECK, this time with \( \kappa[1] = D \) and \( \kappa[2] = B \), to find proofs of \( A \rightarrow D \).

The modification to step 3 is that we do not call COUNT-CHECK for any element in the proof which appears in \( \kappa \). If the proof uses \( B \in T \), and \( B = \kappa[i] \) for some \( i \neq 2 \), then we treat \( B \) as if the checking process had returned a ‘Looped’ result. If \( B = \kappa[2] \) then the status of \( B \) is given by the \( \sqsubseteq \) ordering and \( \kappa[1] \) (i.e., the element being checked by this invocation of COUNT-CHECK); if \( \kappa[1] \sqsubseteq B \) then it is marked as ‘Lose’, if \( B \sqsubseteq \kappa[1] \) then it is marked as ‘Win’. If the two are unrelated it is marked as ‘Looped’: the unresolved contest thus detected is either reported at this stage, or remembered for later reporting (since it may not affect the overall result of the computation). Continuing the example, any \( B \) used in the proof of \( A \rightarrow \neg D \) will be marked ‘Win’ if \( B \sqsubseteq D \), ‘Lose’ if \( D \sqsubseteq B \), and ‘Looped’ otherwise.

To give an example of the computation, take \( T \) containing the following:
CHAPTER 5. COUNTERFACTUALS

\[ p_1 \text{ vehicle}(x) \land \text{park}(y) \rightarrow \neg \text{allowed-on}(x, y) \]

\[ p_2 \text{ ambulance}(x) \rightarrow \text{vehicle}(x) \]

\[ p_3 \text{ ambulance}(x) \land \text{park}(y) \rightarrow \text{allowed-on}(x, y) \]

\[ p_4 \text{ park}(p) \]

For the moment we will assume that none of these elements are related by the 'c' ordering. Let us ask the question ‘\text{ambulance}(a) \rightarrow \text{allowed-on}(a, p)’ (‘If a were an ambulance, would it be allowed on p?’). The steps are as follows:

1 Construct a proof of \( T \vdash \text{ambulance}(a) \rightarrow \neg \text{allowed-on}(a, p) \) using \( p_3 \) and \( p_4 \).

2 Call COUNT-CHECK for \( p_3 \) with \( \kappa = \langle p_3 \rangle \). This involves the following steps:

2.1 Construct a proof of \( T \vdash \text{ambulance}(a) \rightarrow \neg p_3 \) using \( p_1, p_2 \) and \( p_4 \).

2.2 \( T \not\vdash \text{ambulance}(a) \rightarrow \neg p_2 \) and \( T \not\vdash \text{ambulance}(a) \rightarrow \neg p_4 \) so each of these will be marked as ‘Win’ at this level. So consider the call to COUNT-CHECK for \( p_1 \), with \( \kappa = \langle p_1, p_3 \rangle \). The only proof of \( T \vdash \text{ambulance}(a) \rightarrow \neg p_1 \) uses \( p_3 \); but \( p_3 = \kappa[2] \) and so instead of calling COUNT-CHECK recursively for this occurrence of \( p_3 \) we mark it as ‘Looped’ (since \( p_3 \) and \( p_1 \) are unrelated). The call to COUNT-CHECK for \( p_1 \) itself returns a ‘Looped’ result, and at this point we can report the unresolved contest between \( p_1 \) and \( p_3 \).

3 The call to COUNT-CHECK at step 2 will also return ‘Looped’, and since there is no proof of \( T \vdash \text{ambulance}(a) \rightarrow \neg p_4 \), this is the result of the whole computation.

If we now add the information \( p_3 \sqsubseteq p_1 \) to the system (indicating that \( p_3 \) is to beat \( p_1 \) in contests), at Step 2.2 the occurrence of \( p_3 \) will be marked ‘Win’; the call to COUNT-CHECK for \( p_1 \) will return ‘Yes’ (marking \( p_1 \) as ‘Lose’), and the call for \( p_3 \) at Step 2 will return ‘No’ (marking \( p_3 \) as ‘Win’). The result of the computation as a whole will be ‘Yes’.
5.6.2 The SLCC Model

This model - 'SLC with Checking' - extends the SLC formulation of Sec. 4.2.4. We will assume first that all conditional goals are counterfactual in nature and secondly that they are restricted to the first degree; i.e., they have the form \( A \rightarrow B \) where \( A \) and \( B \) are zero-degree. For elementary assertions we will restrict ourselves to atoms and first-degree implications. Together, these restrictions mean that the database will never have more than two segments. We will also only deal here with the propositional case (no variables, no quantifiers and no substitutions).

SLCC states extend their SLC counterparts by having three new elements:

- The status indicator \( \Sigma \), which may take any of the values 'win', 'lose' or 'doubt'.
- The check-tree \( \tau \) which is either empty, or an SLCC tree (as defined below).
- The checks indicator \( \kappa \) which is a list of elementary assertions.

**Definition 28** The initial state for a goal \( G \), an assertion \( A \) (where \( G \) and \( A \) are restricted as above) and a given \( \kappa \) is the state \( ([G]^1, D, U, \epsilon, \Sigma, \tau, \kappa) \) where

- \( D \) contains the single segment \( A \).
- \( U \) is either \( \{1\} \) or \( \emptyset \), depending on whether or not \( t \in A \).
- \( \Sigma \) is set to 'win'.
- \( \tau \) is NULL.

**Definition 29** An SLCC-tree for \( G \), \( A \) and \( \kappa \) is defined as for the SLC-trees of Chapter 4:

- Each node of the tree is an SLCC state.
- The root node is the initial state for \( G \), \( A \) and \( \kappa \).
• Each node has as its descendants all the nodes derivable from it. (The definition of 'derivable' in this context is given below).

**Definition 30** A successful SLCC derivation is cotenable if the $\Sigma$ indicator of every state in the derivation is set to 'win'. A successful or infinite derivation is uncotenable if some state has a $\Sigma$ with the value 'lose'.

**Definition 31** Let $\tau$ be an SLCC-tree. If $\tau$ contains a cotenable branch it is a refuting tree, and if every successful or infinite branch in it is uncotenable it is a supporting tree. Otherwise, it is an indifferent tree.

To simplify the development, from now on we will consider only reduced SLCC trees; given an SLCC tree $\tau$, the corresponding reduced tree results by removing all infinite branches from $\tau$, and from all the check trees recursively contained within its states (see the definition of SLCC states above).

Although we have yet to define the precise notion of an SLCC derivation we can make some general comments about the structure of SLCC trees. In general, the states in the tree will have embedded SLCC trees, and so on recursively. We will define the depth of such an embedded tree in the following straightforward way:

• The depth of the outer-most tree is 0.

• If the depth of a tree is $n$, then the depth of any check-tree in any state in that tree is $n + 1$.

Consider an SLCC tree $\tau$. If there is an $n \in \omega$ s.t. no SLCC tree nested within $\tau$ has a depth greater than $n$, then the extended structure with $\tau$ as its outermost tree is finite. This follows by a straightforward application of Koenig's Lemma, given that we are restricting ourselves to reduced SLCC trees.

Finally, we have the definition of derivability with respect to these new structures:

**Definition 32** Let $\sigma_1$ and $\sigma_2$ be two SLCC states. Then $\sigma_2$ is derivable from $\sigma_1$ if the following hold:
The relationship between the database and goal elements of the two is as for derivability in the SLC model.

Both have the same $\kappa$.

If the state was derived from the previous state in a way which did not use an element of the first segment $S_1$ in $D$ to rewrite the previous goal, or if $k = 1$, indicating that the computation at this point is not directed towards the reduction of a conditional goal, then $\tau$ is NULL and $\Sigma$ is ‘win’.

Otherwise $k = 2$, indicating that the computation at this point is involved in the reduction of a conditional goal, and the state was derived using an element $C$ of the top-level segment $S_1$. At this point $D$ includes two segments, $S_1$ and $S_2$. The new elements are determined according to the following conditions:

- If $\kappa$ has at least two elements, and $C = \kappa[2]$ then $\tau$ is empty and $\Sigma$ is determined by the $\sqsupseteq$ ordering. If $C \sqsupseteq \kappa[1]$ then $\Sigma$ is set to ‘win’; if $\kappa[1] \sqsupseteq C$ then it becomes ‘lose’; otherwise it is ‘doubt’. Otherwise:
  
- If $C = \kappa[j]$ for some $j \neq 2$ then $\tau$ is NULL and and $\Sigma$ becomes ‘doubt’.

Otherwise:

- $\tau$ is the SLCC tree for $S_2 \rightarrow \neg C$, $S_1$ and $\kappa'$ where $\kappa'$ results from $\kappa$ by adding $C$ to the front of it. If $\tau$ is a refuting tree, then $\Sigma$ is set to ‘lose’; if it is a supporting tree then $\Sigma$ becomes ‘win’; otherwise $\Sigma$ is set to ‘doubt’.

**Definition 33** The primary SLCC tree for $T$ and $G$ is the SLCC tree for $A$, $G$ and $\kappa$ where $\kappa$ is NULL.

It is a straightforward matter to verify that SLCC trees satisfy the depth requirement for finiteness given above (given finite $T$); at each increase in depth $\kappa$ is increased by an element of $T$ not already in it, so $\kappa$ must always be finite.

The conjectured relationship between the contest-division formulation and the
SLCC model is as follows: Let \( \langle \text{Win}, \text{Lose} \rangle \) be the contest division for a given \( T, A \) and \( \sqsubset \), and let \( \tau \) be the primary SLCC tree for \( T \) and \( A \rightarrow C \). Then:

- **Win** \( \vdash A \rightarrow C \) iff \( \tau \) contains a cotenable branch.
- **T - Lose** \( \not\vdash A \rightarrow C \) iff every non-failure branch of \( \tau \) is uncotenable.

If the conjecture holds, then the computation is sound with respect to the contest division formulation. For let \( \tau \) contain a cotenable branch. Then \( \text{Win} \vdash A \rightarrow C \), and so \( T' \vdash A \rightarrow C \) for every \( T' \in \text{WIN}(A, T) \) and so \( T \vdash A \rightarrow C \). If every non-failure branch of \( \tau \) is uncotenable then \( T - \text{Lose} \not\vdash A \rightarrow C \), hence \( T' \not\vdash A \rightarrow C \) for every \( T' \in \text{WIN}(A, T) \) and so \( T \not\vdash A \rightarrow C \). In the case where the contest division partitions \( T \) we will also have completeness.

### 5.7 Generalizing the Model

Until this point we have been concerned with \( A \)-restrictions of sets (more precisely, conjunctions) of elementary assertions. How should does this model extend to the case of \( A \)-restrictions of **multisets**? The most straightforward answer is to let

\[
\text{REST}(A, [T_1 \ldots T_m]) = \{[T'_1 \ldots T'_n] \mid T'_1 \in \text{REST}(A, T_1) \ldots T'_n \in \text{REST}(A, T_m)\}
\]

For example, say our database is \( T \) and the query is \( A \rightarrow B \rightarrow C \) (which is of course allowed by the general syntax). Applying the Ramsey Test once, we have:

\[
T \vdash A \rightarrow B \rightarrow C \iff \text{Cot}(A, T), \ A \vdash B \rightarrow C.
\]

Applying it again,

\[
T \vdash A \rightarrow B \rightarrow C \iff \text{Cot}(B, \text{Cot}(A, T)), \ Cot(B, A) \vdash B \rightarrow C
\]

Given the suggested definition, the multiset to the left of the ‘\( \vdash \)’ here is an element of \( \text{REST}(B, \{\text{Cot}(A, T), A\}) \). This certainly looks right, at first glance. However an example will demonstrate that it is probably not what we want. Let \( T \) consist of the following:
Let us ask the question: $A ightarrow B ightarrow D$. On the suggestion this will succeed iff $\text{Cot}(B, \text{Cot}(A, T)), \text{Cot}(B, A), B \vdash D$. But $\text{Cot}(A, T)$ is just $T$ itself; there is no $E \in T$ s.t. $T \vdash A \rightarrow E$. So $\text{Cot}(B, \text{Cot}(A, T)) = \text{Cot}(B, T)$. Similarly, $\text{Cot}(B, T) = T$, and we also have $\text{Cot}(B, A) = A$. So the query succeeds iff $T, A, B \vdash D$ is provable, which of course it is. We take it as intuitively obvious that there is however a contest between p2 and p3 in this situation. Specifically, it seems better to stipulate that the query should succeed iff $\text{Cot}(A \land B, T), \text{Cot}(B, A), A \vdash D$ is provable. Note that $\text{Cot}(A \land B, T)$ will have to give up one of p2 and p3, matching the intuition that there is a contest between these elements.

The formalization of this intuition requires some rearrangement of our structures:

**Definition 34** A restricted assertion has the form $A|B$ where $A$ and $B$ are both (simple) assertions.

**Definition 35** A restricted sequent has the form $\alpha \vdash_R A$ where $A$ is a query and $\alpha$ is a multiset of restricted assertions.

**Definition 36** Let $\alpha$ be a multiset of restricted assertions $[A_1|B_1 \ldots A_m|B_m]$ and $C$ a query. Then $\alpha \vdash_R C$ iff $[\text{Cot}(A_1, B_1) \ldots \text{Cot}(A_m, B_m)] \vdash C$ is provable.

Our new formalization of the Ramsey Test is then:

**Definition 37** $[A_1|B_1 \ldots A_m|B_m] \vdash_R C \rightarrow D$ iff $[C \land A_1|B_1 \ldots C \land A_m|B_m, \text{NULL}|C] \vdash_R D$.

To return to the example, we start off with the database $T$ which we represent by the restricted assertion $\text{NULL}|T$. The question is whether $\text{NULL}|T \vdash R A \rightarrow . B \rightarrow D$ is supported. By two applications of the modified Ramsey Test, this reduces to
the question of \([B \land A|T, B|A, NULL|B] \vdash_R D\), ie, to the provability of \(Cot(A \land B, T), Cot(B, A), B \vdash D\), as required.

The corresponding extensions to the SLCC model follow straightforwardly. Let \(\sigma\) be a state derived using \(B \in S_i\) (where \(S_i\) is some segment in the database) to rewrite a goal with index \(l\) (indicating that the computation at this point is working on a subproof with \(S_i\) as assumptions). Let \(S_i, S_{i-1} \ldots S_i\) be the section of the branch of the database tree containing \(S_i\) and \(S_i\). If \(S_i = S_i\) then the check-tree \(\tau\) is NULL and \(\Sigma\) is set to ‘win’; the restricted assertion corresponding to \(S_i\) in the sequent formulation is \(NULL|S_i\). Otherwise, \(\tau\) is the SLCC tree for the query \(S_i \land \ldots \land S_{i+1} \rightarrow B\) from \(S_i\), with \(\kappa\) as in the original treatment.

### 5.8 First Order Considerations

We will only touch very briefly on these here. The issues are very similar to those involved with exception mechanisms for sound negation-as-failure (see, for example, [Kun87]). As an example, consider the database \(T\) containing:

\begin{align*}
p1 & \quad f(x) \land g(x) \rightarrow h(x) \\
p2 & \quad f(a) \rightarrow \neg g(a) \\
p3 & \quad g(w)
\end{align*}

Let \(p2 \square p3\): interpret this as short-hand for \(p2 \square p3'\) for every instance \(p3'\) of \(p3\). If we put the question \(f(x) \rightarrow g(x)\) to this database, the answer we would intuitively expect is ‘Yes - for every \(x\) except \(a\)’. On the other, if instead of \(p2\) we had

\begin{align*}
p2' & \quad f(x) \rightarrow \neg g(x)
\end{align*}

then we would expect the answer to be ‘Fail’. A more complicated situation arises if we have the following instead of \(p1\):

\begin{align*}
p1' & \quad f(x) \land g(y) \rightarrow h(x)
\end{align*}
Restoring quantifiers, \( p_1' \) is equivalent to \((\forall x)(f(x) \land (\exists y)g(y) \rightarrow h(x))\). Given that we have \((\forall x)g(x)\) in the database, we would expect the query to succeed (for every \( x \)) without qualification, regardless of whether we have \( p_2 \) or alternatively \( p_2' \). For any \( b \) there is a \( c \) s.t. \( f(b) \land g(c) \rightarrow h(b) \) and \( g(c) \) both follow from the database and \( f(b) \rightarrow \neg g(c) \) does not. (Note that there is nothing in our development which restricts us to the Herbrand base - \( b \) and \( c \) may be any terms in the language.)

This last example illustrates the major complication for building the computation up to the full first-order level. Consider the state \( \sigma \) in the depth 0 derivation which was derived using \( p_3 \). The obvious extension to the propositional SLCC model will have us construct a check-tree \( \tau \) for the query \( f(x) \rightarrow g(z) \) where \( z \) is the renaming of the \( w \) in \( p_3 \) applied when \( p_3 \) was used in the derivation. The initial state of \( \tau \) will contain the substitution \( \theta \) associated with \( \sigma \). This \( \tau \) (given that the database contains \( p_2 \) rather than \( p_2' \)) will contain a successful (and, we can assume, cotenable) branch with the constructed substitution \( \theta \rho \) where \( \rho = \{a/x, a/z\} \). But this does not constitute a valid exception to the substitution \( \{x'/x, w'/w\} \) constructed (using just SLC-type moves) in the depth 0 derivation. That is, the complete answer to the query is not ‘Everything except \( a \)’ but rather an unqualified ‘Everything’. The problem is that instead of \((\exists x)(\exists z)(f(x) \rightarrow \neg g(z))\), we should instead be asking \((\exists x)(\forall z)(f(x) \rightarrow \neg g(z))\), as the discussion above makes clear. And this of course is not part of our restricted syntax, having a universal quantifier in a query position. Presumably the solution to the problem relies on apparatus to detect these implicitly existential variables and appropriate Skolemization moves; this is an area of current investigation.

### 5.9 Consistency Maintenance

The same type of strategy may be adapted to the problem of consistency maintenance.

Let us define a new provability relationship as follows:

**Definition 38** Where \( \alpha \) is a multiset of assertions, and \( A \) is a query, \( \alpha \vdash_c A \) if
Cons(α) ⊨ A, where Cons(α) is the consistent part of α.

Where α ⊨ C A, we will say that A is a consistent consequence of α.

To define Cons(α), we will take the same approach that we used for Cot(A, α). First consider the case where α is a single element multiset [T]. We will say that Cons(T) is some self-restriction of T, defined as follows:

**Definition 39** The set of self-restrictions of T, \( \text{SREST}(T) \) is the set of all \( T' \) obeying the following: \( C \in T' \) iff \( C \in T \) and \( T' \vdash \neg C \).

That is, a self-restriction of T contains all and only those elements of T which it does not refute.

**Theorem 97** \( \text{SREST}(T) = \text{REST}(t, T) \)

The proof uses the following theorems of \( LR^+ \):

- \( t \rightarrow A \rightarrow A \)
- \( A \rightarrow_t A \)

So \( T \vdash \neg C \) iff \( T \vdash t \rightarrow \neg C \), and the result is immediate.

In the same way that we had two versions of the Ramsey Test, we could take the following alternative definition of consistent consequence:

\[ T \vdash_C A \text{ iff } T' \vdash A \text{ for all } T' \text{ in some preferred subclass of } \text{SREST}(T). \]

Note that we have the following:

**Theorem 98** \( T \vdash_C A \text{ iff } T \vdash t \rightarrow A \text{ iff } t |_T \vdash R A. \)

For the extension to the general case, we will take it that consistency maintenance is only to apply to the 'top-level' segment. For example, \( T \vdash_C A \rightarrow B \) will be
provable just where \( t | T \vdash_R A \rightarrow B \) is; this reduces in turn to the question of 
\[ [A \land t | T] \vdash_R A \rightarrow B, \] 
then \([A \land t | T, A] \vdash_B B\) and finally \([\text{Cot}(A \land t, T), A] \vdash B\) (rather than \([A \land t | T, t | A] \vdash_R B\) and \([\text{Cot}(A \land t, T), \text{Cot}(t, A)] \vdash B\)). We take it that where the question is \( A \land \neg A \rightarrow B\), for example, then there is no reason to enforce consistency with respect to the segment formed from \( A \land \neg A\); the question is ‘Would \( B \) follow if \( A \) were inconsistent?’.

The details of the extensions to the SLCC model for the consistent consequence case are straightforward. Assume \( B \in S_1 \) (where \( S_1 \) is the ‘top-level’ segment) is used to reduce a goal with index \( l \). Let \( S_1, S_{l-1} \ldots S_1 \) be the branch of the database tree containing \( S_i \) and \( S_1 \). Then the check-tree \( \tau \) in the state derived using \( B \) is the SLCC tree for the query \( t \land S_2 \land \ldots \land S_l \rightarrow \neg B \) from \( S_1 \).

5.10 Discussion

In [Gin86] Ginsberg gives a good description of a counterfactual system based on the Stalnaker-Lewis theory of conditionals ([Sta68],[Lew73]). Loosely, we can characterize the approach as follows: Let \( \text{MAX}(A, T) \) be the set of \( T' \subseteq T \cup \{A\} \) which are maximally consistent (in the sense that \( T' \cup \{B\} \) is inconsistent for any \( B \in T \cup \{A\} - T' \)) and which also contain \( A \). If \( A \) is contradictory then we let \( \text{MAX}(A, T) \) consist of a single \( T' \) which contains all the sentences in the language. We have an ordering on elements of \( \text{MAX}(A, T) \), which we write as \( T_1 \leq T_2 \), which we interpret as saying that \( T_1 \) is at least as similar to \( T \) as \( T_2 \) is. Let \( \text{SIM}(A, T) \) be the set of minimal elements of \( \text{MAX}(A, T) \) under this ordering. This gives the following version of the Ramsey Test:

\[ T \vdash A \rightarrow C \text{ iff } T' \vdash C \text{ for every } T' \in \text{SIM}(A, T) \]

Again loosely, the difference between the Lewis approach on the one hand, and the Stalnaker approach on the other, is that in the latter we stipulate that the ‘\( \leq \)’ ordering is such that \( \text{SIM}(A, T) \) is always a singleton. This syntactic description is only a paraphrase of the Stalnaker-Lewis theory, which is in fact worlds-semantic in nature, but it will do for our purposes.
We could, if we liked, reformulate this to take account of relevance and an inconsistency tolerating negation, as follows: Say that $T$ is $A$-consistent if $T \not\models A \rightarrow \lnot B$ for any $B$ s.t. $T \models B$. Let $\text{MAX}'(A, T)$ be the set of maximally $A$-consistent subsets of $T$. Let $\text{SIM}'(A, T)$ be the set of $T' \in \text{MAX}'(A, T)$ s.t. $T' \cup \{A\}$ is minimal under the $\leq$ ordering. Then $T \models A \rightarrow C$ if $T', A \models C$ for all $T' \in \text{SIM}'(A, T)$.

The essential differences between this kind of approach, and the Goodman-style one we have taken here, are as follows:

- We talk in terms of $A$-restrictions rather than maximally $A$-consistent subsets.
- We have an operationally-defined $\sqsubseteq$ ordering on formulas rather than the similarity ordering $\leq$ on sets.

Of the two, the first is not essential; we could have recast the development of this Chapter in terms of $A$-consistency without changing any of the important features. For the other, we can if we like view our $\sqsubseteq$ ordering as inducing a similarity ordering $\leq$: for all $T_1 \in \text{WIN}(A, T)$ and $T_2 \in \text{REST}(A, T)$ set $T_1 \leq T_2$, so that $\text{SIM}'(A, T) = \text{WIN}(A, T)$.

There are clear advantages for practical applications of allowing rankings on sets of formulas to be induced by rankings on the individual elements. The alternative, in the worst case, would force us to explicitly list all of the possible subsets that might arise and the rankings between them (Ginsberg’s system in [Gin86] has various restrictions on the generality of the $\leq$ ordering to avoid this). So for example in Rescher’s plausible reasoning system ([Res76]) the ranking on sets is induced by a plausibility ranking on individual formulas, and we see the same kind of thing in the various Bayesian developments in the expert systems area. The major problem with this approach is independence; for example, while it might be true that both $B$ and $C$ are ‘better than’ $C$ according to some scale of values (probability, plausibility, similarity etc.) there is in general no reason to suppose that the conjunction, $A \land B$, is better than $C$. And so sets $T_1$ and $T_2$ cannot generally be ranked just by local, pair-wise considerations of the relationships between their individual elements; to decide whether some $B$ is in
the ‘best’ of the possible alternatives we will in the worst case again be forced to generate the alternatives in full. Our operationally-defined □ ordering avoids these independence problems, since it is not based on any context-dependent theory of value.

5.11 Implementation

CLOGPROG’s user-interface has a switch COUNT/NOCOUNT. When COUNT is set, all implications in query positions are interpreted as counterfactuals. Later versions will allow explicit distinction between indicatives and counterfactuals in the syntax. It also has a switch CONSIS/NOCONSIS; when CONSIS is set all answers will be in terms of consistent consequence. To rank elements, we allow labels on assertions in the database, and we have a non-logical command

\texttt{rank(label1, label2)}

the effect of which is to set \( A \sqsubseteq B \) whenever \( A \) and \( B \) are elementary assertions in the database labelled with \( label1 \) and \( label2 \) respectively.

The computational details, for the propositional case, follow fairly straightforwardly from the models presented here. The first order aspects are still under development.

We also have two additional switches, NAF/NONAF and RULEPREF/NORULEPREF. The first of these is coupled to CONSIS/NOCONSIS, so that setting NAF also sets CONSIS, and setting NOCONSIS sets NONAF as well. When NAF is set, the effect is to implicitly add an instance \( \neg p \) to the database, for all atomic sentences \( p \), and to adjust the \( \sqsubseteq \) ordering so that this added instance is beaten by everything. Operationally, this approximates the closed world assumption; if any computation can use \( \neg p \) it will succeed, unless \( p \) follows from the database, because \( p \) will win the contest which results. This is of course not a viable way of implementing negation as failure (since the resulting computations are very expensive) but it is interesting for experimental purposes. The other switch, RULEPREF/NORULEPREF, is used to implicitly alter the \( \sqsubseteq \) ordering so that for any atom \( p \) and any implication \( A \rightarrow q \) in the database, \( A \rightarrow q \sqsubseteq p \) unless we already have \( p \sqsubseteq A \rightarrow q \). This allows us to easily model those common situations
in which we want rules to hold more strongly than facts.
Bibliography


BIBLIOGRAPHY


BIBLIOGRAPHY


