The fact that logic cannot satisfy us awakens an almost insatiable hunger for the irrational.
- A.N Wilson

PROOF SYSTEMS FOR PROPOSITIONAL MODAL LOGIC

by

THELMA DAWN VAN DER VYVER

submitted in part fulfilment of the requirements
for the degree of

MASTER OF SCIENCE

in the subject

COMPUTER SCIENCE

at the

UNIVERSITY OF SOUTH AFRICA

SUPERVISOR: MR T.A. MEYER

November 1997

ABSTRACT

In classical propositional logic (CPL) logical reasoning is formalised as logical entailment and can be computed by means of tableau and resolution proof procedures. Unfortunately CPL is not expressive enough and using first order logic (FOL) does not solve the problem either since proof procedures for these logics are not decidable. Modal propositional logics (MPL) on the other hand are both decidable and more expressive than CPL. It therefore seems reasonable to apply tableau and resolution proof systems to MPL in order to compute logical entailment in MPL. Although some of the principles in CPL are present in MPL, there are complexities in MPL that are not present in CPL. Tableau and resolution proof systems which address these issues and others will be surveyed here. In particular the work of Abadi & Manna (1986), Chan (1987), del Cerro & Herzig (1988), Fitting (1983, 1990) and Gorȩ (1995) will be reviewed.

Keywords: Classical propositional logic, propositional modal logic, resolution proof systems, tableau proof systems, local logical entailment, global logical entailment.

160

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Disciples should be on their guard against the seductions of words and sentences and their illusive meanings, for by them, the ignorant and dull-witted become entangled and helpless as an elephant floundering around deep mud. Words and sentences .. cannot express highest reality . The ignorant and simple minded declare that meaning is not otherwise than words, that as words are, so is meaning. .. Truth is beyond letters and words and books.

- The Tao Te Ching (LXXXI)

1. HISTORY

Throughout the ages psychologists have researched the acquisition of language. Many theories and approaches have been developed as a result of this research. Fundamentally the human species progresses through three stages of language acquisition. During the first stage they develop an understanding of their language, learning how to distinguish between different individual sounds of their language. Pre-linguistic speech follows shortly afterwards during which they attempt to imitate these sounds. Finally they enter the linguistic speech stage where the human learns not only to utter one-word sentences but also grammatically correct verbal sentences. Once people know how to utter correct sentences they will focus on the context of a given sentence in order to understand the meaning of the sentence, that is, most people are aware that the same sentence may have different meanings in different circumstances. For example the phrase: "I am going to kill you!" will alarm people if uttered by an adversary. However, if the same phrase is uttered by an acquaintance, the phrase may mean that "I will be very upset if you pull the plug before I have saved my work on the PC!"

Although these stages describe language acquisition amongst children in all countries, we could use the same approach to describe the acquisition of a second language. For instance, consider an English speaking person who wants to learn a different language, say French. This person will have to expose his/her ear to the sounds of the French language, learn to imitate and give meaning to the different sounds, and eventually learn how to form grammatically correct sentences in French such as: 'Bonjour, madame' (Good morning madam).

As with language acquisition, logic can also be developed along similar stages. Firstly, for every logical language we need to define the alphabet and punctuation symbols that we can use in the language. Once we know what the given language looks like we generally define rules that allow us to construct words for the language, and in turn we use these words to construct sentences for our language. Finally we define different contexts in which we can use the given sentences which allows us to assign meaning to these sentences. In contrast to language acquisition, we progress one step further in logic and use our sentences in their given contexts to derive new sentences. This process of derivation is known as automated reasoning.

Artificial Intelligence is a field in computer science that concerns itself not only with tasks that require intelligence, but also tasks that can be performed on a computer. Since the ability to make logical deductions is considered to be an integral part of human intelligence automated reasoning, or theorem proving, has created widespread interest in Artificial Intelligence.

The origins of theorem proving can be traced back as far as 1656 when Gottfried Wilhelm von Leibniz first devised a general decision procedure to prove theorems. In 1930 Herbrand introduced a mechanical theorem prover for a branch of logic known as first order logic (a logic that makes use of notions such as terms, predicates and quantifiers). What made this theorem prover noteworthy was that the theorem prover was based on an algorithm that would halt after a finite number of trials. The advantage of having a finite algorithm was that it was now possible for the first time in history to automate a proof procedure. This is exactly what Gilmore did in 1960, when he implemented Herbrand's procedure on a computer.

With the advent of computers it became more and more important to find theorem provers that could be implemented efficiently. It was in this light that J.A. Robinson's breakthrough in theorem proving in 1965 was so important. He developed a theorem prover that used a single inference rule, known as the resolution principle. Compared to other theorem provers at the time, his theorem prover showed remarkable efficiency. Since then much research has gone into improving the resolution principle with regards to efficiency and automation. (Chang & Lee 1973.)

Aristotle (384-322 B.C) was perhaps one of the first investigators to explore logical relationships between the necessary, the impossible, the possible and the permitted. C.I. Lewis in 1918 initiated the modern symbolic analysis of modality and Kurt Gödel in 1932 put forward a modal logic based on the ordinary propositional calculus (see chapter 2) with the schemata K (see chapter 3) and the rule of Necessitation. Shortly after Robinson's breakthrough G.E. Hughes published his book, 'Introduction to Modal Logic.'

From 1959 to 1980 much research and thought went into modal logic which finally culminates in an axiomatic account of modal logics. During this period a young genius, Saul Kripke, laid down the foundations of modern propositional and predicate modal logic in several influential papers. His notion of a frame (see page 37) is probably his most noteworthy contribution in the field of modal logics.

Somewhere between 1980 and 1988 the penny finally dropped. It's all very well to have a logic that concerns itself with expressions of various 'modes' in which statements may be true, but unfortunately the axiomatisations of these logics are very difficult to apply and are extremely time consuming. Since the life span of a logician is limited (and time costs money) there was clearly a need for more efficient theorem provers. It is therefore not surprising that a number of logicians decided to follow in the footsteps of Robinson with the development of a number of resolution theorem provers for modal logics.

As with resolution, tableaux theorem provers were also initially designed for the automation of reasoning specifically for classical logics. Historically, tableaux systems have been divided into two fundamental groups, that of syntactic tableaux and semantic tableaux. Semantic tableaux systems have found prominence in fields of automated deduction, i.e. fields in which the primary emphasis is on finding proofs. Syntactic tableaux systems, on the other hand, have found prominence in fields of type theory where the main emphasis has been on the ability to distinguish between different proofs in order to place computational interpretations on the proofs. Gerhard Gentzen was responsible for most of the work done in syntactic tableaux round about 1935. Curry appears to be the first logician to extend Gentzen's systems to modal logic. Beth on the other hand has been responsible for much of the original work in semantic tableaux, with Kripke once again extending his work to modal logics. Zeman in 1973 gave an account of both syntactic and semantic systems and in 1983 Fitting published his well known work in tableaux systems which provides an exposition of tableau systems for most of the basic logics. (Gorè 1995.)

Although modal logic was originally only of interest to philosophers it has recently had many applications in the fields of computer science and artificial intelligence. In particular some of these applications have been in temporal logics, formal program specifications, concurrency, hardware verifications (Gorè 1995) and nonmonotonic logics in artificial intelligence.

2. SCOPE

Due to the effective application of resolution- and tableau- based proof systems in classical propositional logic as well as that of first order logic it seems reasonable to extend these systems to that of modal propositional logic. While some of the ideas of these proof systems, such as the use of refutation procedures, are still applicable, some of the advantages of these proof procedures do not carry over directly to propositional modal logic. For instance, the conversion of formulas into an appropriate normal form is difficult to attain as well as restricting the number of inference rules to a bare minimum.

This thesis surveys attempts by a number of researchers to capture various aspects of these resolution and tableau proof systems in modal logic. In particular the focus will be on how these researchers use their systems to compute logical reasoning formalised as logical entailment. The systems surveyed here are those of Abadi & Manna (1986), Chan (1987), del Cerro & Herzig (1988), Fitting (1983, 1990) and Gorè (1995). The description of these systems will focus on the three specific modal systems, KT4, KT4.3 and KT5, because they seem to be of real interest in Artificial Intelligence.

The foundation for these topics are laid in chapters two and three. Chapter 2, covers the concepts of propositional logic and some of its respective approaches to theorem proving namely: axiomatic systems, resolution and tableaux proof systems. Note that although natural deduction and its applications in propositional logic is important it is not within the scope of this thesis. Chapter 3, builds upon the foundation of propositional logic and provides a synopsis of modal logic as well as introducing the reader to the normal systems of modal logic KT4, KT4.3 and KT5. Chapters 4 and 5 constitute the main focus of this thesis and present the resolution and tableaux proof systems for modal logic as described by the different researchers. Chapter 6 concludes with suggestions to alternative applications of these modal resolution and tableaux proof systems to other branches of artificial intelligence in particular that of nonmonotonic logic.

CHAPTER 2

CLASSICAL PROPOSITIONAL LOGIC

Interpreting a symbol is to associate it with some concept or mental image, to assimilate it to human consciousness.

- PJ Davis & R Hersh

1. SYNTAX

When learning any language such as English we have to familiarise ourselves with the words in the language. Not only do we have to learn how to recognise the words but we also have to learn how to spell them correctly so that people from different walks of life can recognise that when we write 'cat' we mean a four legged creature that meows. As our learning progresses we learn how to combine the different words into meaningful sentences by means of the connectives such as *not*, *if*.. *then*, *and*, and *or*. Eventually, after much study, we are able to write a simple sentence that most people can relate to, for example: *The cat sat on the mat.*

A propositional language is a precisely defined formal language that can be acquired in much the same way as any other written language. We start off by defining the words in our language and what they look like. We refer to the words in the propositional language as 'symbols' and they can be represented by any one of the following:

- A possible infinite set of atoms represented by the letters P, Q, R ....
- The propositional connectives: $\neg$, $\rightarrow$, $\wedge$, $\vee$, $\leftrightarrow$.
- The constants $\top$, $\bot$.

Informally we can define the above symbols by means of the following table. A more precise description of these symbols will follow in section 2.1.

Symbol	**Meaning**
$\neg$	not
$\rightarrow$	if ... then
$\wedge$	and
$\vee$	or
$\leftrightarrow$	if and only if
$\perp$	false
$\top$	true

The items listed in the table, together with the atoms, make up the words of the language, from which sentences can be built up. However, as with the English language, we need to know what the rules are for forming sentences or formulas. These rules are defined below (Chang & Lee 1973:7):

Definition 1

***Well-formed formulas**, or formulas for short, in propositional logic are defined recursively as follows:*

1. *An atom is a formula*
2. *$\top$ and $\perp$ are formulas*
3. *If A is a formula, then $(\neg A)$ is a formula.*
4. *if A and B are formulas, then $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$ and $(A \leftrightarrow B)$ are formulas.*

All formulas are generated by applying the above rules.

Note that the symbols '(' and ')' are punctuation symbols and are not words in this language.

Using this definition, we can formulate formulas in the propositional language. Say, for example, that we would like to formalise the statement: "Odie is my dog and Garfield is my cat". Firstly we need to define our atoms. Let O stand for "Odie is my dog" and let G stand for "Garfield is my cat" (note that we could have used any atom, we just chose O and G for convenience). If we refer back to definition 1 (1) we will find that we now have two formulas namely O and G. However, we would like to connect these two formulas. So we refer to definition 1 to see if there is a connective that can help us join these two formulas. Looking closely at statement 4 in definition 1, and referring to the informal meaning of $\wedge$ above, we notice that the connective $\wedge$ will do the trick. So now we can join our two formulas to produce $(O \wedge G)$.

We can summarise these steps by looking at another example: "If Garfield is a cat then Odie is not a cat"

1. Define the atoms of our formula:
   Let G be "Garfield is a cat"
   Let O be "Odie is a cat"
   i.e. We now have "if G then not O".

2. Replace the connectives with the correct symbols using definition 1.
   $G \rightarrow \neg O$.

The reading of the formula $(G \rightarrow \neg O)$ is quite straightforward, but this is not always the case when we deal with more complex formulas such as $(G \wedge O \vee P)$. Without the use of punctuation symbols or precedence rules it will be very difficult to determine whether this formula should be read as $((G \wedge O) \vee P)$ or $(G \wedge (O \vee P))$. At the same time the longer our formulas become, the more cumbersome the use of brackets will become. It is therefore useful to assign precedence rules to the connectives so that we may eliminate some of the brackets without creating ambiguity. We will assign the following precedence to our connectives in decreasing rank: $\leftrightarrow$, $\rightarrow$, $\wedge$, $\vee$, $\neg$, requiring that the connective with greater rank always reaches further. Thus $(G \wedge O \vee P)$ will mean $(G \wedge (O \vee P))$.

2. SEMANTICS

2.1 Interpretations

Semantics is related to meaning in language. For example what is the semantics of the sentence: *Garfield is mad*? In other words, what do we mean when we say that *Garfield is mad*? Are we referring to his mental capabilities, in which case we mean *Garfield is insane* or are we making a statement about his emotions, i.e. *Garfield is annoyed*? Generally in the English language, we can determine the meaning of a sentence in terms of the context in which the sentence was uttered. In other words, it is often very difficult to interpret a sentence in isolation.

We have a similar phenomenon in propositional logic, in that we need to be able to assign meaning to propositional formulas. A formula such as $P \vee Q$ has no meaning unless we understand what P and Q represent and in what context they are being used. In propositional logic we use the word *interpretation*, to refer to the context in which a formula is used.

Definition 2

Given a propositional formula A, let A_1, A_2, ..., A_n be the atoms occurring in the formula A. Then an ***interpretation*** *of A is a function from the set of atoms to the set {T, F}, where T represents true and F represents false (Chang & Lee 1973:9).*

It is only when we are given an interpretation of the language that we can say something about the meaning of the formulas. Generally in propositional logic when we refer to the 'meaning of a formula' we mean that we would like to determine if the formula is true or false in a given interpretation. This begs the question, what do we mean by: *A formula is true/false in an interpretation?*

Definition 3

The ***truth*** *of any well formed formulas in an interpretation I is recursively defined as follows:*

1. *For every atom A, A is true in I if and only if I(A) = T. (A is false in I if and only if I(A) = F).*
2. *For any well formed formula A, ¬A is true in I if and only if it is not the case that A is true in I.*
3. *For any two well formed formulas A and B, A ∧ B is true in I if and only if A is true in I and B is true in I.*
4. *For any two well formed formulas A and B, A ∨ B is true in I if and only if either A is true in I or B is true in I or both.*
5. *For any two well formed formulas A and B, A → B is true in I if and only if either A is false in I or B is true in I, or both.*
6. *For any two well formed formulas A ↔ B is true in I if and only if A and B have exactly the same truth value in I (i.e. either both A and B are true in I or both are false in I). (Chang & Lee 1973:8.)*

For example, the propositional logic language containing just the two atoms P and Q has four distinct interpretations. We can denote these interpretations by I_1, I_2, I_3 and I_4 represent them as follows:

- $I_1(P) = T$ and $I_1(Q) = T$
- $I_2(P) = T$ and $I_2(Q) = F$
- $I_3(P) = F$ and $I_3(Q) = T$
- $I_4(P) = F$ and $I_4(Q) = F$

To determine if any well formed formula is true in a particular interpretation say I_2, we simply use the definition above. For instance, take the formula (P ∨ Q). In I_2 P gets the value T and Q gets the value F. From definition 3 (4) we see that this means that (P ∨ Q) will get the value T in I_2.

Note that for any interpretation I, and any formula A, A will either be true or false (but never both) in I.

Definition 4

A formula is said to be ***valid*** *if and only if it is true under all its interpretations. A formula is said to be* ***invalid*** *if and only if it is not valid (Chang & Lee 1973: 11).*

In propositional logic, we refer to valid formulas as *tautologies*. We say that a tautology is a formula that is true under all its interpretations. Tautologies also have the added advantage that their semantics can be determined from their logical structure. For example, for any formula A, (A ∨ ¬A) is a tautology, that is, for any interpretation I, (A ∨ ¬A) will always be true in I.

Definition 5

A formula is said to be ***unsatisfiable*** *if and only if it is false under all its interpretations. A formula is said to be* ***satisfiable*** *if and only if it's not unsatisfiable. An interpretation I* **satisfies** *a formula if and only if the formula is true in I and it satisfies a set of formulas Γ if and only if I satisfies each formula A ∈ Γ (Chang & Lee 1973:11).*

By convention if a formula A is unsatisfiable then we refer to A as a *contradiction.* A propositional formula A is a contradiction if and only if it is false under all its interpretations. For example (P ∧ ¬P) is a contradiction as this formula can never be true for any interpretation I, since for any interpretation I, both P and ¬P will have to be true and this is impossible in propositional logic.

Definition 6

For any two formulas A and B, A is ***logically equivalent*** *to B if and only if (A ↔ B) is true for every interpretation I.*

2.2 Semantic Entailment

In the previous sections we defined the syntax and interpretations of well formed formulas. In this section the semantics of the propositional language is taken one step further in order to define the idea of one formula following logically from a set of other formulas.

Definition 7

A set of formulas Γ ***logically entails*** *a formula A (written Γ □ A) if and only if every interpretation I that satisfies all the formulas in Γ also satisfies the formula A.*

To illustrate the above definition. Let Γ = {(P → Q), ¬Q} and A be the formula ¬P. To show that Γ logically entails A i.e. {(P → Q), ¬Q} □ ¬P, we need to show that every interpretation I that satisfies Γ also satisfies ¬P. One way of determining whether Γ logically entails A is to use a truth table. Consider the truth table for Γ ∪ {A}:

P	**Q**	**P → Q**	**¬Q**	**¬P**
T	T	T	F	F
T	F	F	T	F
F	T	T	F	T
F	F	T	T	T

From the truth table we can see that any interpretation that evaluates all formulas in Γ to true also evaluates ¬P to true. We can therefore safely say that Γ □ ¬P.

If the set Γ is the empty set (denoted ∅) then the only formulas that are entailed by Γ are the set of tautologies. In other words ∅ □ A if and only if A is a tautology. This means that if the empty set logically entails A then A is a tautology and if A is a tautology then A is logically entailed by the empty set. Generally we will write ┐ A instead of ∅ □ A. For example for the tautology (P ∨ ¬P) we have □ (P ∨ ¬P).

Logical entailment can also be defined in terms of validity and unsatisfiability.

Theorem 1

Given a set of formulas $\Gamma = \{ \Gamma_1, \Gamma_2, ..., \Gamma_n\}$ and a formula A, Γ logically entails A if and only if $(\Gamma_1 \wedge \Gamma_2 \wedge ... \wedge \Gamma_n) \rightarrow A$ is valid.

(See Chang & Lee 1973:16 for the proof.)

For example if we want to show that ¬P is logically entailed by {(P→Q), ¬Q} we need to show, by theorem 1, that (P → Q) ∧ ¬Q → ¬P is valid. We do so by means of the following truth table.

P	**Q**	**P → Q**	**¬Q**	**¬P**	**(P → Q) ∧ ¬Q → ¬P**
T	T	T	F	F	T
T	F	F	T	F	T
F	T	T	F	T	T
F	F	T	T	T	T

From the truth table we can see that $(P \rightarrow Q) \wedge \neg Q \rightarrow \neg P$ always evaluates to T regardless of the truth values of the other columns, i.e. $(P \rightarrow Q) \wedge \neg Q \rightarrow \neg P$ is true under all its interpretations and by definition 4 valid. It follows from theorem 1 that we can conclude that $\neg P$ is logically entailed by $\{(P \rightarrow Q), \neg Q\}$.

Alternatively, instead of using validity to show logical entailment we could use unsatisfiability (see definition 5), that is, in order to show that a formula A is logically entailed by a set of formulas Γ we can show that $\Gamma \cup \{\neg A\}$ is unsatisfiable. We record this result in the following theorem.

Theorem 2

Given a set of formulas $\Gamma = \{ \Gamma_1, \Gamma_2, ..., \Gamma_n\}$ and a formula A. Γ logically entails A if and only the set $\{\Gamma_1, \Gamma_2, ..., \Gamma_n, \neg A\}$ is unsatisfiable.

(See Chang & Lee 1973:16 for proof.)

For example, if we want to show that $\neg P$ is logically entailed by $\{(P \rightarrow Q), \neg Q\}$ we need to show, by theorem 2, that $\{(P \rightarrow Q), \neg Q, \neg\neg P\}$ is unsatisfiable. We do so by means of the following truth table.

P	**Q**	**P → Q**	**¬Q**	**¬P**	**¬¬P**
T	T	T	F	F	T
T	F	F	T	F	T
F	T	T	F	T	F
F	F	T	T	T	F

From the truth table we can see that there does not exist a row which contains all T's, that is, there is no interpretation that satisfies $\{(P \rightarrow Q), \neg Q, \neg\neg P\}$ and by definition 5 it is therefore unsatisfiable. It follows from theorem 2 that we can conclude that $\neg P$ is logically entailed by $\{(P \rightarrow Q), \neg Q\}$.

Note that from the results of theorem 1 it is sufficient in propositional logic to only look at the sets of tautologies when we want to determine logical entailment. However, this result does not hold for logics such as modal logic which we discuss in the next chapter.

In the above example we were fortunate that Γ was a small finite set of sentences. However, in propositional logic, we don't always have this luxury. We often have to deal with infinite or very large sets of sentences. Clearly, when Γ is infinite we can't use truth tables to show logical entailment as we did in the above examples. It would therefore be useful to have a result ensuring that we need only concern ourselves with finite sets of sentences. The following theorem provides us with this result.

Theorem 3 (Compactness Theorem)

Given an infinite set of formulas Γ, and a formula A, Γ □ A if and only if there is a finite subset Γ_0 of Γ such that Γ_0□ A.

(See Fitting 1990b:55 for proof.)

3. PROOF PROCEDURES

Thus far our study of propositional logic has focused on the syntax and semantics of propositional formulas. In this section we will extend our study to the analysis of the process of deduction.

Informally a *proof* can be defined as a demonstration that a conclusion follows from a set of well formed formulas. A *proof procedure* on the other hand is defined as a procedure for finding proofs. The main aim of these proof procedures is therefore to determine whether a given formula A is a either logically entailed by a set of formulas Γ or not logically entailed by Γ. In other words we use proof procedures to determine whether or not it is the case that Γ□ A.

It is instructive to evaluate why we would even bother to go to all the effort of designing proof procedures when using truth tables, as shown above, would appear to do the trick. One apparent difficulty with truth tables is that in order to establish that a formula A is logically entailed by a set of formulas Γ we have to verify that each interpretation I that satisfies Γ also satisfies A. The problem is that the number of interpretations that satisfies Γ may be infinite. This means that it would be impossible to verify that A was logically entailed by Γ by means of truth tables in a finite amount of time. Since we would like to use the computer to verify the validity of A in Γ we are clearly going to need an alternative technique to solve this problem, as it is not practical to tie a PC up for an unlimited amount of time. Fortunately, there is an important theorem in logic that states that whenever a set of formulas Γ logically entails A, there is a finite proof of A from Γ. Hence the question of determining logical entailment is reduced to the problem of finding such a proof. (Genesereth & Nilsson 1987.)

In this section three types of proof systems are described namely: Hilbert proof systems, resolution and tableaux systems. A brief exposition of each procedure is provided, as well as a description of how these different proof procedures determine logical entailment.

3.1 Hilbert Style Systems

The first proof procedures that we will look at were designed by Hilbert in the 1920's. These proof procedures are generally referred to as Hilbert style systems. There are many strains of Hilbert systems but if one was to analyse all the systems one would be able to extract the following common characteristics from these systems,. that is, all Hilbert systems have (Hughes 1968):

1. A set of primitive symbols.
2. A set of formation rules specifying which formulas are to count as well formed formulas.
3. A selected set of well formed formulas, known as axioms.
4. A set of transformation rules licensing various operations on the axioms and well formed formulas obtained by previous applications of the transformation rules. (Rules of transformation are often referred to as *inference rules*).

Note that item 1 and 2 make up the symbols and the well formed formulas of the language. In this instance the set of primitive symbols and formation rules are those of the propositional language as defined in sections 1 and 2 of this chapter.

To illustrate these concepts we will briefly look at a Hilbert system defined by Hamilton (1988):

- An alphabet of symbols:
  $\neg, \rightarrow, (,), P, Q, R \ldots$

- A set of well formed formulas defined recursively as:
  1. Each atom P, Q, R ... is a well formed formula.
  2. If A and B are well formed formulas then $(\neg A)$ and $(A \rightarrow B)$ are well formed formulas
  3. The set of all well formed formulas is generated by 1 and 2 above.

- Axioms, specified in terms of axiom schemes (i.e. in terms of sentence structures).
  1. $(A \rightarrow (B \rightarrow A))$
  2. $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$
  3. $(((\neg A) \rightarrow (\neg B)) \rightarrow (B \rightarrow A))$

- Rules of deduction (also defined in terms of schemes):
  Modus Ponens: From A and $(A \rightarrow B)$, B is a direct consequence, where A, B are any well formed formulas of the system.

The discerning reader will notice that a number of the connectives that were listed in section 1 are not used by Hamilton in this example. The reason for this is that Hamilton defines the connectives defined in section 1 by means of the adequate set of connectives ¬ and →. (Refer to Hamilton 1988:19 for the proof.) For example the formula $A \wedge B$ can be defined in terms of the connectives ¬ and → to obtain the equivalent formula $\neg(A \rightarrow \neg B)$.

Definition 8

*A **proof in a Hilbert system** from a set Γ of formulas is a finite sequence A_1, A_2, ..., A_n of formulas such that each formula is either an axiom, or is a member of Γ or follows from applying the rules of inference to well formed formulas earlier in the sequence. We call a formula A, a **theorem** of Γ, if A is the last line of a proof from Γ, i.e. if $A = A_n$. We will write Γ□ A to symbolise that A has a proof from Γ in the Hilbert system (Hamilton 1973:29).*

A special case of the above definition is when Γ is the empty set, that is $\Gamma = \varnothing$. For this case we generally just write □ A instead of ∅□ A.

We will use the Hilbert system, as defined by Hamilton above, to illustrate this definition. Assume that we want to use this system to show that Q is logically entailed by $\{P, P \rightarrow Q\}$.

Looking at the definition of Hamilton's system it should be clear that there are three axiom schemas and one rule of inference that can be used to prove Q. Using definition 8 as the modus operandi the proof is as follows:

1. P Given
2. $P \rightarrow Q$ Given
3. Q from 1 and 2 using Modus Ponens.

Since Q is the last line of our proof we can conclude by definition 8 that Q is a theorem relative to $\{P, P \rightarrow Q\}$.

Note that it is customary, as demonstrated in the example, to display proofs by writing one formula per line. Generally, we will also indicate how we have derived the formulas by making notes to the right of the formulas.

3.2 Resolution

Hilbert systems, as described above, can be inefficient since we often have to consider all the axioms and inference rules in the system in order to prove that a formula A is a theorem of a set of formulas Γ. In other words, for large Hilbert systems it is not always a trivial task to construct a proof for A from Γ. For example, try to find a proof for □ (P → P).

Since the advent of computers there has been a number of attempts to improve the efficiency of theorem proving procedures in order to facilitate the automation of these procedures. One of these attempts was a proof system introduced by Robinson in 1965 referred to as resolution. What made this proof procedure so attractive at the time, was that it was simple to analyse and implement (Genesereth & Nilsson 1988). What added to the efficiency of the proof procedure was that it only made use of one very efficient inference rule known as the resolution principle (defined below).

In this section we will provide a brief overview of the basics of resolution. Advanced resolution strategies such as semantic resolution, linear resolution, unit resolution are not within the scope of this dissertation and the interested reader is referred to works of authors such as Chang & Lee (1973) and Genesereth & Nilsson (1987) for more details.

The notion of a *refutation system* is required for the remainder of this section. We therefore digress for a moment and describe briefly what a refutation system is. Essentially a refutation system is a proof system that uses given axioms and inference rules to obtain a proof of a formula A from a set of formulas Γ. The proof is obtained by negating A and using the predefined axioms and inference rules to derive a contradiction from Γ ∪ {¬A}. If we succeed in deriving a contradiction then we can safely conclude that A is logically entailed by Γ. In other words since Γ ∪ {¬A} is unsatisfiable we can conclude by theorem 2 that Γ logically entails A. Note that different refutation proof systems will provide different definitions of what a contradiction is in their systems as well as the means by which these contradictions are derived.

In the English language resolution is defined as a separation into elements, decomposition, conversion into another form. Effectively this is what resolution is about in propositional logic. It hinges on the fact that we can decompose propositional formulas into sets of formulas. Once the transformation has been completed, the transformed set can be used to construct other sets of formulas. The proof procedure then involves showing how we can use this process of transformation to eventually derive a proof in terms of sets of formulas.

Before we describe this process in more detail, we need to define what we mean by *clauses* and in order to do so we need to understand what *literals* are.

Definition 9

A ***literal*** *is an atomic sentence or* ***the*** *negation of an atomic sentence. An atomic sentence is a* ***positive literal****, and the negation of the atomic sentence is a* ***negative literal****. If A is an atom then the two literals A and ¬A are said to be each other's* ***complement*** *(Genesereth & Nilsson 1987:64).*

Definition 10

A ***clause*** *is a set of literals.*

In propositional logic the symbol '$\wedge$' is often referred to as a *conjunction.* Similarly the symbol '$\vee$' is referred to as a *disjunction.* A formula is in *conjunctive normal form* if it has the following structure: ($(A_{11} \vee A_{12} \vee \ldots \vee A_{1n}) \wedge (A_{21} \vee A_{21} \vee \ldots \vee A_{2m}) \wedge \ldots \wedge (A_{k1} \vee A_{k2} \vee \ldots \vee A_{ks})$), where the A_{ij}'s are literals. These terms will be used in the following discussion.

Due to the nature of resolution proof systems formulas are generally converted to clausal form in order to reduce the computational complexity of the formulas and to provide a uniform notation. This conversion from a well formed formula to clausal form is accomplished by means of the following algorithm. The algorithm provides a sequence of rules that can be applied to any well formed formula in order to convert it into its logically equivalent clause form, that is, the semantics of the initial formula remain intact during the conversion process.

Conversion to clausal form (Genesereth & Nilsson 1987)

1. Remove all the $\rightarrow$, $\leftrightarrow$, $\leftarrow$ from the formula using the following rules:

1.1 $A \rightarrow B$ is replaced by $\neg A \vee B$

1.2 $A \leftrightarrow B$ is replaced by $(\neg A \vee B) \wedge (A \vee \neg B)$

2. Replace T and $\perp$ as follows:

2.1 T is replaced by $(P \vee \neg P)$

2.2 $\perp$ is replaced by $(P \wedge \neg P)$

3. Distribute the negation over the formula, until there is only one atom in the scope of every ¬, using the following rules:

3.1 $\neg\neg A$ is replaced by A

3.2 $\neg(A \vee B)$ is replaced by $\neg A \wedge \neg B$

3.3 $\neg(A \wedge B)$ is replaced by $\neg A \vee \neg B$

4. Place the expression in conjunctive normal form using the following rules repeatedly

4.1 $A \vee (B \wedge C)$ is replaced by $(A \vee B) \wedge (A \vee C)$

4.2 $(B \wedge C) \vee A$ is replaced by $(B \vee A) \wedge (C \vee A)$

5. Eliminate the connectives using the following rule:

5.1 $(A_{11} \vee A_{12} \vee \ldots \vee A_{1n}) \wedge (A_{21} \vee A_{22} \vee \ldots \vee A_{2m}) \wedge \ldots \wedge (A_{k1} \vee A_{k2} \vee \ldots \vee A_{ks})$ is replaced by $\{A_{11}, A_{12}, \ldots A_{1n}\}, \{A_{21}, A_{22}, \ldots A_{2m}\} \ldots \{A_{k1}, A_{k2}, \ldots A_{ks}\}$

Conversely the finite set of clauses $\{\{A_{11}, A_{12}, \ldots A_{1n}\}, \{A_{21}, A_{22}, \ldots A_{2m}\} \ldots \{A_{k1}, A_{k2}, \ldots A_{ks}\}\}$ can be rewritten as the following well formed formula $(A_{11} \vee A_{12} \vee \ldots \vee A_{1n}) \wedge (A_{21} \vee A_{22} \vee \ldots \vee A_{2m}) \wedge \ldots \wedge (A_{k1} \vee A_{k2} \vee \ldots \vee A_{ks})$

Note that a clause represents the disjunction of its literals and occasionally clauses will be treated as if they are formulas.

Let's illustrate this routine with a simple example say $\neg(P \leftrightarrow Q)$.

1. **Remove the ↔.**
Rule 1.2 is used to convert
$\neg(P \leftrightarrow Q)$ to $\neg((\neg P \vee Q) \wedge (P \vee \neg Q))$.

2. **Distribute the negation.**
Rule 3.3 is used to convert
$\neg((\neg P \vee Q) \wedge (P \vee \neg Q))$ to $\neg(\neg P \vee Q) \vee \neg(P \vee \neg Q)$.
Then rule 3.2 is used to convert
$\neg(\neg P \vee Q) \vee \neg(P \vee \neg Q)$ to $(\neg\neg P \wedge \neg Q) \vee (\neg P \wedge \neg\neg Q)$.
Finally the formula is refined using rule 3.1 to get: $(P \wedge \neg Q) \vee (\neg P \wedge Q)$

3. **Place the formula in conjunctive normal form.**
Thus $(P \wedge \neg Q) \vee (\neg P \wedge Q)$ becomes $((P \wedge \neg Q) \vee \neg P) \wedge ((P \wedge \neg Q) \vee Q)$
Refining this formula further yields:
$(P \vee \neg P) \wedge (\neg Q \vee \neg P) \wedge (P \vee Q) \wedge (\neg Q \vee Q)$.

4. **Eliminate the connectives.**
Hence $(P \vee \neg P) \wedge (\neg Q \vee \neg P) \wedge (P \vee Q) \wedge (\neg Q \vee Q)$ can be reduced to the following clauses:
$\{P, \neg P\}$ $\{\neg Q, \neg P\}$, $\{P, Q\}$ and $\{\neg Q, Q\}$

We now have clauses but we still have no idea how to use them in proving anything. This is where we need to define the resolution principle (Genesereth & Nilsson 1987).

Definition 11 (Resolution Principle)

For any two clauses C_1 and C_2, if there is a literal L_1 in C_1 that is complementary to a literal L_2 in C_2, then delete L_1 and L_2 from C_1 and C_2 respectively, and construct the union of the remaining clauses. The constructed clause is called a ***resolvent*** *of C_1 and C_2.*

For instance, from the example above the two clauses {P, Q} and {¬Q, Q} will be resolved with each other since Q is in the one clause and ¬Q is in the other, thus giving {P, Q} as resolvent.

The notion of a *proof* in a resolution system follows from the following definition:

Definition 12

A ***unit clause*** *is a clause with only one literal. An* ***empty clause*** *is a clause with no literals.*

A proof system using resolution can be described in terms of a refutation system, that is, in order to show that there exists a proof for a formula A from a set of formulas Γ, A is negated and the resolution principle is used to derive a contradiction from Γ ∪ {¬A}. A contradiction is found once two clauses are derived that have the form {A} and { ¬A}. Applying the resolution principle to these two clauses results in the empty clause. In other words, if we succeed in deriving the empty clause we have effectively found a contradiction. We define this process formally by means of the following definition.

Definition 13

Given a set Γ of clauses, a ***resolution*** *(deduction) of A from Γ is a finite sequence $A_1, A_2, \ldots A_k$ of clauses such that each A_i is either a clause in Γ or a resolvent of clauses preceding A_i where $A = A_k$. A deduction of the empty clause from Γ ∪ {¬A} is called a* ***refutation or proof*** *of A from Γ (Chang & Lee 1973:73).*

A is defined as a *theorem* of Γ if the empty clause is deduced from Γ ∪ {¬A}. Hence it follows from this definition that if we want to show that a formula A is a theorem of a set of formulas Γ all we need to do is negate A, convert A and Γ into clausal form and then try to deduce the empty clause.

This definition is illustrated by showing that Q is a theorem of {P → Q, P}. The formulas are converted to clause form using the conversion rules above.

1. P → Q is replaced by {¬P, Q}
2. P is replaced by {P}

From the discussion above it should be clear that in order to prove that Q is a theorem of the set of clauses given in 1 and 2 above, we need to negate Q, write it in clausal form and try to derive the empty clause. Negating Q and writing it in clause form gives us the following clause: {¬Q} and as a result the following set of clauses {{¬P, Q}, {P}, {¬Q}}. Listing these clauses sequentially the final result is proved as follows:

1.	{¬P, Q}	Given
2.	{P}	Given
3.	{¬Q}	Given
4.	{¬P}	From 1 and 3 (Q, ¬Q)
5.	{}	From 2 and 4 (P, ¬P)

In general resolution proofs are shown in a stepwise format as done in the example above. Also note that we could have applied the resolution principle in more than one way. For instance, we could have used the following proof instead of the one given above.

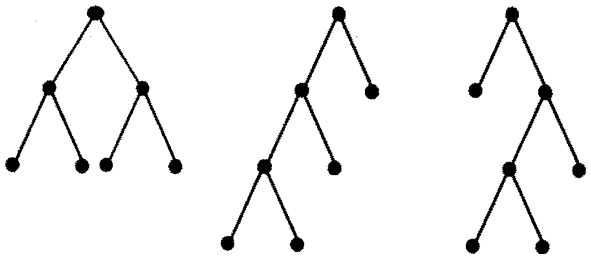
1.	{¬P, Q}	Given
2.	{P}	Given
3.	{¬Q}	Given
4.	{Q}	From 2 and 1 (P, ¬P)
5.	{}	From 3 and 4 (Q, ¬Q)

It follows therefore that resolution proofs are not unique. This phenomena will also be apparent in the resolution systems that are presented in chapter 4.

3.3 Tableaux Systems

As with resolution, tableaux proofs are also refutation systems, that is, in order to prove that A is a theorem of a set of clauses Γ, we negate A and attempt to find a contradiction using a tableau system. This section will therefore provide a brief overview of tableau systems and how they can be used to determine whether a formula A is logically entailed by a set of formulas Γ.

In computer science we often refer to data structures known as trees. These are merely structures that are made up of different nodes that branch to the left or right of a central node. We call a tree structure a *binary tree* if every node has a left and right child (refer to the left most diagram below). Each node is usually assigned a meaning such as, one node for every state, one node for every formula, one node for each class. The branches are used to connect the nodes in order to allow the traversing of the tree in a forward or backward fashion. The shape of the tree is therefore determined by the way the nodes are connected via the branches. For example, a tree with seven nodes may have one or none of the following shapes:



The end nodes of the trees are often referred to as leaf nodes of the tree. The binary tree on the left, for instance, has four leaf nodes.

In artificial intelligence these tree structures are used to construct tableaux proofs. For instance the nodes in the tableaux proofs in this section will be made up of propositional formulas and as the proof advances new nodes and branches will be created. As with resolution our main aim is to find a refutation proof for A from a set of formulas Γ. We prove this result by listing ¬A and the set of formulas Γ sequentially as the branch of a tree. We then use a number of rules to expand the formulas, which in turn expands the tree, until we reach a contradiction. In the rest of this section we will look at what rules we may use to expand a tree in tableaux systems as well as how to recognise a contradiction.

In tableaux systems we refer to the rules used to expand the initial set of formulas as *tableau expansion rules*. For the purpose of this dissertation we will use the tableau expansion rules as described by Fitting (1990b). In order to do so we need to look at two types of formulas referred to as α and β types.

α type formulas are formulas that have the same structure as the formulas that appear on the left side of the following table. These formulas are generally considered as *and* formulas, that is, if the α formula in the left column is true then both the formulas listed under α_1 and α_2 will be true as well. For example, if we are given that $(A \wedge B)$ is true then we can safely assume that A is true and that B is true. The same result holds for the other formulas in the table.

α	α_1	α_2
$(A \wedge B)$	A	B
$\neg(A \vee B)$	$\neg A$	$\neg B$
$\neg(A \rightarrow B)$	A	$\neg B$
$A \leftrightarrow B$	$A \rightarrow B$	$B \rightarrow A$

β type formulas are formulas that have the same structure as the formulas that appear on the left side of the following table. These formulas are generally considered to be *or* formulas, that is, if the β formula is true then either β_1 or β_2 or both will be true. So for instance if we are given that $(A \vee B)$ is true then we can safely assume that A is true or B is true or both. Similarly for the other formulas in the table.

β	β_1	β_2
$(A \vee B)$	A	B
$\neg(A \wedge B)$	$\neg A$	$\neg B$
$(A \rightarrow B)$	$\neg A$	B
$\neg(A \leftrightarrow B)$	$\neg(A \rightarrow B)$	$\neg(B \rightarrow A)$

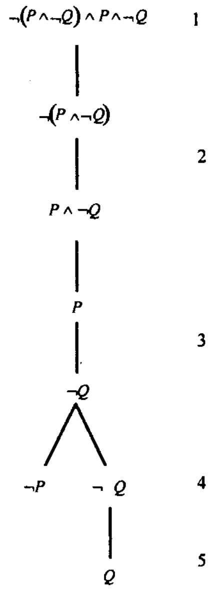
Tableau expansion rules:

current node	**new node**
$\neg\neg A$	A
$\neg T$	$\perp$
$\neg\perp$	T
α	α_1 α_2
β	$\beta_1 \mid \beta_2$

Consider any tree T and any branch R of the tree. We can expand the tree using the tableau expansion rules defined above as follows (Fitting 1990b:37):

- For every occurrence of the formula $\neg\neg A$ on R, extend R by adding a node labelled A to its end.
- For every occurrence of the formula $\neg T$ on R, extend R by adding a node labelled $\perp$ to its end.
- For every occurrence of the formula $\neg\perp$ on R, extend R by adding a node labelled T to its end.
- For every occurrence of the formula α on R, extend R by adding two nodes, one below the other. Label the one node α_1 and the other node α_2.
- For every occurrence of the formula β on R, extend R by adding a left and right node. Label the one node β_1 and the other β_2.

For example, we can transform the formula $(\neg(P \wedge \neg Q) \wedge P \wedge \neg Q)$ into the following tree using the tableau expansion rules:



Although the numbers on the right are not part of the proof we have nevertheless used them to make the following discussion clear. Step 1 in the tree is the root node of the tree and is labelled by the initial formula. Step 2 is the result of applying the α expansion rule to the initial formula. Step 3 is the result of applying the α expansion rule to the formula $P \wedge \neg Q$. Step 4 is the result of applying a β tableau expansion rule to $\neg(P \wedge Q)$. Finally step 5 is the result of applying the first tableau expansion rule, as shown in the table above.

This process of constructing and expanding a tree is formalised in the following definition.

Definition 14

Let $\{A_1, ..., A_n\}$ be a finite set of propositional formulas:

*1. The following one branch tree is a **tableau** for $\{A_1, ..., A_n\}$*

$$\begin{array}{c} A_1 \\ A_2 \\ \dots \\ A_n \end{array}$$

2. If T is a tableau for $\{A_1, ..., A_n\}$ and T results from T by the application of a Tableau Expansion rule then T* is a tableau for $\{A_1, ..., A_n\}$ (Fitting 1990b:37).*

A tableau proof is defined by means of the following two definitions.

Definition 15

*A branch B of a tableau is called **closed**:*

1. if both A and ¬A occur on B for some propositional formula A

2. or if $\bot$ occurs on B.

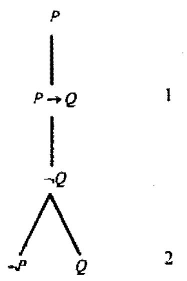
*We say that a **tableau is closed** if every branch B is closed (Fitting 1990b:38).*

Definition 16

*A **tableau proof** of a formula A from a set of formulas Γ, is a closed tableau for $\Gamma \cup \{\neg A\}$. We say that A is a **theorem** of Γ if A has a tableau proof from Γ (Fitting 1990b:38).*

A tableau proof can be demonstrated by means of the following example:

Consider once again the example of Tweety and Garfield. Representing the formulas in terms of P and Q we show that Q is logically entailed by {P, P→Q} by means of a tableau proof. In other words by definition 16, we show that the tableau for $\{P, P \rightarrow Q\} \cup \{\neg Q\}$ is closed. Applying definition 14 to the set of formulas we get the following tableau:



Step 1 is the root of the tableau and contains the initial formulas of our system. Step 2 is the result of applying the β rule to (P → Q). Now by definition 15.1 if A and ¬A appear on the same branch then the branch is closed. If we look at the left hand side of our tree we notice that both P and ¬P appear on the branch hence we can close the left branch. Similarly if we look at the right hand side of the tree we notice that both Q and ¬Q occur on the same branch so we can close this branch as well. Since both branches are closed we can conclude by definition 15 that our tableau is closed. Hence by definition 16 we have a tableau proof for Q from {P, P → Q}.

4. SOUNDNESS AND COMPLETENESS

Before we close this chapter we need to discuss some properties of the three proof systems that we have given above. The first property that we will look at states that if a proof exists for a formula A, from a set of formulas Γ, then we know that A is logically entailed by Γ. This property is known as soundness. The second property that we will look at is known as completeness. This property ensures that if a formula A is logically entailed by a set of formulas Γ then we know that a proof exists for A from Γ.

4.1 Hilbert Proof Systems

Theorem 4 (Hilbert Soundness)

If a proof exists for a formula A from a set of formulas Γ, then A is logically entailed by Γ, i.e. if Γ□ A then Γ□ A.

(See Fitting 1990b:75 for a proof.)

Theorem 5 (Hilbert Completeness)

If a formula A is logically entailed by a set of formulas Γ then there exists a proof for A from Γ, i.e. if Γ□ A then Γ□ A.

(See Fitting 1990b:75 for a proof.)

4.2 Resolution Proof Systems

We will focus on two variations of soundness and completeness for resolution proof systems. Firstly we will look at resolution soundness and completeness for the special case where Γ is the empty set. Secondly we will focus on what is known as *strong* resolution soundness and completeness i.e. resolution soundness and completeness for any sets of formulas Γ.

We state resolution soundness without proof.

Theorem 6 (Resolution Soundness)

If A has a resolution proof, then A is a tautology, i.e. if □ A then □ A.

Theorem 7 (Resolution Completeness)

If A is a tautology then A has a resolution proof, i.e. if □ A then □ A.

Proof: Before proving completeness we require the following definition and theorem (which we state without proof). (See Fitting 1990b for details.)

1. **Definition**

   Γ is ***resolution consistent*** *if and only if (iff) the empty clause cannot be derived from Γ.*

2. **Model Existence Theorem**

   Let Γ be a finite set of clauses. If Γ is resolution consistent, then Γ is satisfiable, i.e. there is an interpretation I that satisfies every C ∈ Γ.

We prove completeness as follows: Suppose that A is a tautology and that it is not the case that □ A, that is, the empty clause cannot be derived from the set {¬A}. It follows by 1 that the set {¬A} is resolution consistent. By 2 this means that {¬A} is satisfiable for some interpretation I. This in turn means that ¬A is true in I. But then A cannot be a tautology, that is, it is not the case that □A. □

Theorem 8 (Strong Resolution Soundness)

For any set of formulas Γ, and any formula A: If Γ□ A then Γ□ A, i.e. if there exists a resolution proof for a formula A from a set of formulas Γ then Γ logically entails A.

(See Fitting 1990b:67 for a proof.)

Theorem 9 (Strong Resolution Completeness)

For any set of formulas Γ, and any formula A: If $\Gamma \square A$, then $\Gamma \square A$, that is, if a formula A is logically entailed by a set of formulas Γ, then there exists a resolution proof for A from Γ.

Proof: The following results, stated without proof, are required in order to prove completeness.

1. **Lemma**

   $\Gamma \square A$ iff there is a finite set $\Gamma_F \subseteq \Gamma$ such that $\Gamma_F \square A$.

2. **Lemma**

   Let $\Gamma_F = \{A_1, A_2, \ldots, A_n\}$. Then $\Gamma_F \square A$ iff $\square A_1 \wedge \ldots \wedge A_n \rightarrow A$.

Completeness is proved as follows: Assume that $\Gamma \square$A. By the compactness theorem (theorem 3) it follows that there is a finite set $\Gamma_F \subseteq \Gamma$ such that $\Gamma_F \square$ A. Let $\Gamma_F = \{A_1, A_2, \ldots, A_n\}$ then by theorem 1, $\square A_1 \wedge A_2 \wedge \ldots \wedge A_n \rightarrow A$ is valid. From resolution completeness it follows that $\square A_1 \wedge A_2 \wedge \ldots \wedge A_n \rightarrow A$. Hence by the lemma in 2 above, we have $\Gamma_F \square$ A, and therefore $\Gamma \square$ A by 1. $\square$

4.3 Tableaux Proof Systems

Similar to resolution, two variations of soundness and completeness for tableaux systems will be presented. In the first variation tableaux soundness and completeness will be presented for the case where Γ is the empty set. In the second variation a stronger version of tableaux soundness and completeness is presented where Γ is any set of formulas.

Theorem 10 (Tableau Soundness)

If A has a tableau proof then A is a tautology, i.e. if $\square A$ then $\square A$.

(Refer to Fitting 1990b:50 for the proof.)

Theorem 11 (Tableau Completeness)

If A is a tautology then A has a tableau proof, i.e. if □A then □A.

Proof: Before we prove tableau completeness we need the following definition and theorem (stated without proof). (Details can be found in Fitting 1990b.)

1. ***Definition.***

   *A finite set Γ of propositional formulas is **tableau consistent** if there is no closed tableau for Γ.*

2. ***Model Existence Theorem.***

   If the set of formulas Γ is tableau consistent then Γ is satisfiable.

We now prove tableau completeness. Assume that A is a tautology and that it is not the case that □A. Then by definition there is no closed tableau for {¬A}. Then by 1 {¬A} is tableau consistent. But then by 2, ¬A is satisfiable. Therefore A can't be a tautology, that is, it is not the case that □ A. ■

Thus far we have been dealing with tableaux, that make use of finite sets of formulas, Γ. As with resolution the set Γ may be infinite. We therefore need to review what the impact of this infinite set may be on tableau proofs. By definition of a tableau proof we know that if the tableau for Γ ∪ {¬A} is closed then A is a theorem of Γ. The problem is how to construct the tableau proof. Clearly, since Γ is infinite, we cannot list all the members of Γ at the top of the tableau as was done previously, since this would take an infinite amount of time. A new approach is therefore required that will permit the construction of a tableau for Γ ∪ {¬A} in a finite amount of time. This tableau can be constructed by using the following Γ-rule as defined by Fitting (1990b).

Definition 17

*The **Γ-introduction rule** for a tableau is: Any member A of Γ, where Γ is an infinite set of formulas, can be added to the end of any tableau branch. We write Γ□ A if there is a closed tableau for {¬A}, allowing the Γ-introduction rule for a tableau.*

Essentially if Γ is infinite we will start constructing a tableau by picking an arbitrary formula from Γ and extend the tableau by means of applying the tableau expansion rules. If we need another formula from Γ we will simply add it to a branch. We will keep repeating this process until we find a proof for some formula A ∈ Γ. From these results we can now state strong tableau soundness and completeness.

Theorem 12 (Strong Tableau Soundness)

For any set of formulas Γ, and any formula A: If $\Gamma \Box A$ then $\Gamma \Box A$, that is, if there exists a tableau proof for a formula A from a set of formulas Γ then A is logically entailed by Γ.

Proof: The following preliminary results are required, which are stated without proof:

1. **Lemma**

   $\Gamma \Box A$ iff there is a finite set $\Gamma_F \subseteq \Gamma$ such that $\Gamma_F \Box A$.

2. **Lemma**

   Let $\Gamma_F = \{A_1, A_2, ..., A_n\}$. Then $\Gamma_F \Box A$ iff $\Box A_1 \wedge ... \wedge A_n \rightarrow A$.

Suppose that $\Gamma \Box A$. Then, by 1 there is a finite set $\Gamma_F \subseteq \Gamma$ such that $\Gamma_F \Box A$. It follows therefore that $\Box A_1 \wedge A_2 \wedge \ldots \wedge A_n \rightarrow A$ by 2 above (where $\Gamma_F = \{A_1, A_2, \ldots, A_n\}$). Hence $\Box A_1 \wedge A_2 \wedge \ldots \wedge A_n \rightarrow A$ by tableau soundness and therefore by theorem 1 $\Gamma_F \Box A$. Hence by the compactness theorem we have $\Gamma \Box A$. □

Theorem 13 (Strong Tableau Completeness)

For any set of formulas Γ, and any propositional formula A: If $\Gamma \Box A$ then $\Gamma \Box A$, that is, if a formula A is logically entailed by a set of formulas Γ, then there exists a tableau proof for A from Γ.

Proof: Suppose $\Gamma \Box A$. Then by the compactness theorem there is a finite set $\Gamma_F \subseteq \Gamma$, such that $\Gamma_F \Box A$. Let $\Gamma_F = \{A_1, A_2, \ldots A_n\}$. Then by theorem 1, $\Box A_1 \wedge A_2 \wedge \ldots \wedge A_n \rightarrow A$. So by tableau completeness, $\Box A_1 \wedge A_2 \wedge \ldots \wedge A_n \rightarrow A$. Then by the lemma from 2 of theorem 12, $\Gamma_F \Box A$. Therefore $\Gamma \Box A$ by the lemma from 1 of theorem 12. □

5. DECIDABILITY

We conclude this chapter with the notion of decidability. In propositional logic we use the term *decidable* to describe a proof procedure that when given a formula A and a set of formulas Γ is guaranteed to terminate and output either a 'yes' (A is a theorem of Γ) or 'no' (A is not a theorem of Γ) answer. By 'yes' we mean that Γ logically entails A and by 'no' we mean that Γ does not logically entail A. We will discuss decidability with regards to resolution and tableau systems. Note that propositional logic is decidable for all finite sets Γ, but only semi-decidable for all sets Γ (i.e. including infinite sets).

5.1 Resolution

To show decidability for resolution we need to describe a proof procedure that meets the following requirements:

- Accepts a formula A as input
- Accepts a finite set of formulas Γ.
- Uses resolution to find a proof of A from Γ, if one exists.
- If there exists a proof for A from Γ then the procedures terminates with a 'yes' answer otherwise the procedure terminates with a 'no' answer.

We will describe one such procedure. This procedure is simplistic and inefficient but nevertheless illustrates the fact that resolution is decidable for finite sets of formulas.

INPUT:

- A formula A
- A finite list of formulas Γ

PROCEDURE:

```
PROCEDURE CONVERT(A)
    A = Implications_out(A)
    A = Replace_⊥(A)
    A = Replace_T(A)
    A = Negations_in(A)
    A = Disjunctions_in(A)
    A = Connectives_out(A)
END CONVERT
```

```
PROCEDURE RESOLVE(Γ)
    i = 1.
    REPEAT1
        IF length(Γ) < i THEN
            Return (Failure)        /* end of clause list no proof found */
        END IF
        Clause1 = ChooseClause(Γ, i)
        j = 1.
        REPEAT2
            IF length(Γ) < j THEN   /* end of list */
                Exit Repeat2
            END IF
            Clause2 = ChooseClause(Γ, j).
            IF Clause1 <> Clause2 THEN
                Resolvent = ResolveClauses(Clause1, Clause2)
                IF Resolvent is not the empty set THEN
                    IF {} is in Resolvent THEN
                        Return(Success)
                    ELSE
                        Concatenate(Γ, Resolvent).
                    END IF.
                END IF
            END IF
            j++
        END REPEAT2
        i++
    END REPEAT1
END RESOLVE
```

```
PROCEDURE START

    A = Negate(A).
    Concatenate ($\Gamma$, A).

    Convert($\Gamma$)   /* for example after conversion $\Gamma = \{\{P\}, \{\neg P,Q\}, \{\neg Q, R\}, \{\neg R\}\}$ */
    Resolve($\Gamma$)

    IF Success THEN
      Return(Yes)
    ELSE
      Return(No)
    END IF.

END START
```

OUTPUT:

Yes (there exists a proof for A from Γ) or *No* (there is no proof for A from Γ) answer.

The procedure Implications_out systematically removes all the implications from the formulas in Γ, using a similar process to that defined in the conversion procedure (item 1) in section 3.2. This procedure will eventually terminate since Γ is finite and therefore there are only a finite number of formulas that need to be converted. Similar arguments hold for the remaining of the procedures listed in the procedure CONVERT. Replace_$\bot$ and Replace_T follows the same procedure as that of the conversion procedure item 2. Item 3 of the conversion procedure is performed by Negations_in, item 4 by Disjunctions_in and finally item 5 is performed by the procedure Connectives_out. It should therefore be clear that since Γ is finite each of the procedures listed in the procedure CONVERT will take a finite amount of time and as a result the procedure CONVERT will also take a finite amount of time and will eventually terminate with a finite list of clauses.

In the procedure RESOLVE the only other procedure that is listed is the procedure ResolveClauses. Essentially this procedure compares the literals in the two clauses with each other in order to ascertain if there are any complementary literals. Once two complementary literals have been identified their resolvent is derived. Note that for the sake of completeness it is assumed that this procedure will resolve all complementary literals. In other words the variable Resolvent may contain more than one clause, if there was more than one way to resolve the two clauses. For example if the clauses given to the procedure ResolveClauses are $\{\neg P, Q\}$ and $\{\neg Q, P\}$ then Resolvent $=\{\{Q, \neg Q\}, \{\neg P, P\}\}$ at the end of the procedure. Although this is not the most appropriate way to handle this dilemma it does avoid a lot of backtracking later on in the procedure. Clearly the procedure ResolveClauses will eventually terminate since there are only a finite number of literals in the two clauses and thus there is only a finite number of ways to resolve the literals with each other.

The procedure RESOLVE systematically works through all the clauses in Γ attempting to resolve them with each other. Sooner or later all the resolvents of the clauses will have been added to Γ since there are only a finite number of clauses that can be resolved with each other. It follows therefore that this procedure will terminate in a finite amount of time and as a result the procedure START will also terminate in a finite amount of time.

5.2 Tableaux

To show decidability for tableaux systems with finite sets of formulas we need to describe a proof procedure that meets the following requirements:

- Accepts a formula A as input
- Accepts a finite set of formulas Γ as input.
- Uses a tableau proof to find a proof of A from Γ, if one exists.
- If there exists a tableau proof for A from Γ then the procedure terminates with a 'yes' answer otherwise the procedure terminates with a 'no' answer.

Once again there are many different procedures that implement decidability for tableaux systems. We will only look at one such example. Although inefficient, it nevertheless works. This procedure can be described by means of the following pseudo code:

Note that we will assume that if a node has only one child node then the child node will be referred to as the left node.

INPUT:

- A formula A.
- A finite set of formulas $\Gamma = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_n\}$

PROCEDURE:

Create the initial branch of the tree and list all the input formulas sequentially, that is $\neg A, \Gamma_1, \Gamma_2, \ldots, \Gamma_n$.

Number the tableau expansion rules sequentially.

current node = root node.

End of tree = false.

```
WHILE not End of tree

    FOR each tableau expansion rule, $R_i$.
        Expand the tree by applying the rule $R_i$ to the formula at the Current node
    END-FOR.

    New node found = false.
    REPEAT
        IF current node has an unmarked left child THEN
                current node = left child node.
                New node found = true.
        ELSE IF current node has a unmarked right child THEN
                current node = right child node.
                New node found = true.
        ELSE
                mark current node.
                IF current node = root node THEN
                        End of tree = true.
                ELSE
                        current node = parent node.
                END-IF.
        END-IF.
    UNTIL New node found OR End of tree.

END WHILE
```

```
FOR each branch in the  tree
    IF two formulas A and $\neg$A appear on the branch or $\perp$ THEN
        Mark the branch as closed.
    END-IF.
END-FOR.

IF all the branches are marked closed THEN
    End the procedure with a 'yes' answer
ELSE
    End the procedure with a 'no' answer.
END-IF.
```

OUTPUT:

A 'yes' if $\Gamma \cup \{\neg A\}$ is unsatisfiable else 'no' .

This procedure expands the tableau tree by applying the expansion rules to each of the formulas at a given node. The expansion rules are applied to all the nodes on the left side of the tree and then to the nodes on the right side of the tree. Sooner or later all the expansion rules will have been applied to all the formulas at all the nodes, since Γ is finite. At this stage all we need to do is determine whether the branches of the tree are all closed or not. This will involve traversing each branch and determining whether there are two propositional formulas of the type A and $\neg$A. Once we have completed this task we have to see whether we have a closed tableau or not. If we do, then we know that there exists a proof for A from Γ, and from tableau soundness and completeness we know-that Γ logically entails A. If the tableau is not closed we know that there does not exist a proof for A from Γ and therefore we can conclude that it is not the case that Γ logically entails A.

6. CONCLUSION

In this survey of classical propositional logic the emphasis has been on highlighting the techniques developed by researchers to permit proofs to be generated in computer environments. From chapter 3 onwards theses ideas and strategies will be extended to propositional modal logic.

CHAPTER 3

PROPOSITIONAL MODAL LOGIC

It was a doorway to magic, mysterious, brain cracking worlds, worlds where you had to tread carefully, worlds where you made up your own rules, worlds where you had to accept complete responsibility for your actions.

- Fynn

1. INTRODUCTION

Modal logic, an extension of pure propositional logic, is an ancient subject devised by philosophers and modelled by mathematicians. Roughly speaking, it allows us to define the modes in which a proposition could be true or false. For example, the statement 'Garfield is on the mat' is an assertion of the indicative. The same statement could on the other hand be expressed in a different mood as the more conditional: 'Garfield might be on the mat' (i.e. it is possible that Garfield is on the mat), and even more forceful: 'Garfield must be on the mat' (i.e. it is necessary that Garfield is on the mat).

To facilitate the expression of phrases such as 'it is possibly the case that' and 'it is necessary that' we introduce two unary propositional connectives $\Diamond$ and $\Box$. The unary connective $\Box$ expresses various modes of truth such as:

- it is necessarily true that A;
- it has always been true that A;
- it is known that A;
- it is provable that A;
- throughout the computation, A;
- along all future paths, A.

The unary connective $\Diamond$ expresses modes of truth such as:

- it is possible / could be / might be that A is true;
- probably A;
- at the next state, A;
- it will eventually be true that A;
- it ought to be that A.

We call these unary connectives $\Box$ and $\Diamond$ modal operators and the systems of logic in which they are contained are called modal systems or modal logics. These systems are used extensively in various branches of artificial intelligence and computing sciences such as logics of knowledge and belief, logics of programs and for tasks such as the specification of distributed and concurrent systems. For example, for a set S of possible states of a computation process we can define a relation R on S such that sRt means that there is an execution of a non-deterministic program that starts in state s and terminates in state t. Then we could define $\Box A$ to mean that 'every terminating execution of the program brings about A', and $\Diamond A$ to mean that the program enables A, i.e. 'there is some execution that terminates with A true.'

This chapter provides a brief introduction to modal logic based primarily on the work of Chellas (1980). Section 2 introduces the syntax of propositional modal logic with the semantics of propositional modal logic presented in section 3. In section 4 we will introduce normal systems of modal logic and in particular we will define the normal systems KT4, KT4L and KT5. Section 5 covers soundness and section 6 completeness of these normal systems. Section 7 shows determination for these systems and section 8 concludes this chapter with a discussion on decidability for the normal systems KT4, KT4L, KT5.

2. PROPOSITIONAL MODAL LOGIC

This section is devoted to a recital of the basic syntactic concepts for the language of modal logic.

The language is founded on:

- A possible infinite set of *atomic formulas* represented by the letters P, Q, R ....
- The logical connectives: $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\Box$, $\Diamond$.
- The constants T, $\perp$
- The punctuation symbols "(" and ")".

To aid readability we will assign the following order of precedence among the connectives: $\Box$, $\Diamond$ $\neg$, $\wedge$, $\rightarrow$, $\leftrightarrow$.

The set of formulas of propositional modal logic can be defined recursively as follows:

Definition 18

1. *Any atom A is a formula.*
2. *T is a formula.*
3. *⊥ is a formula.*
4. *If A is a formula then ¬A, □A, ◊A are formulas.*
5. *If A and B are formulas then A ∧ B, A ∨ B, A → B, A ↔ B are formulas.*
6. *All formulas are generated by applying the rules above.*

3. STANDARD MODELS

3.1 Definition

Recall that in chapter 2 we assigned meaning to the formulas in the propositional language in terms of interpretations. In modal logic we use standard models to represent a notion similar to that of interpretations in propositional logic. Standard models therefore determine the semantics of modal logic formulas.

Definition 19

M = <W, R, V> is a ***standard model*** *if and only if (iff):*

1. *W is a set of possible worlds.*
2. *R is a binary relation on W (i.e. R ⊆ W x W).*
3. *V is a mapping of atoms to subsets of W*
   (i.e. V(P) ⊆ W for an atom P).(Chellas 1980:68.)

The term 'model' and 'standard model' will be used interchangeably in this dissertation.

For example M=<W, R, V> where:

- W = {α, β},
- R = {<α,α>, <β,β>} and
- V(P) = {α}
- V(Q) = {β}

is a standard model, where P and Q are the only atoms in the language.

The reader may encounter the notion of a *frame* in alternative sources of modal logic. A frame is simply the model M defined above without the mapping V, that is, F = <W, R> is a frame.

3.2 Truth

Now that we have defined what a standard model looks like we need to define when a formula is true in a given standard model and when it is not. From the definition above it should be clear that standard models consist of a set of different worlds. Each of these worlds provide a different interpretation for a given formula. It is therefore possible for a formula, say A, to be true at one world and false at another, or for A to be true at all the worlds in a given model, or none of the worlds. We will write $\square_{\alpha}^{M}$ A to mean that A is true at the world α in a model M.

The truth conditions for statements in a standard model are defined as follows (Chellas 1980:35):

Definition 20

Let α be a world in a standard model M = <W, R, V>.

1. *For any atom P, $\square_{\alpha}^{M} P$ iff $\alpha \in V(P)$*
2. $\square_{\alpha}^{M} T$.
3. *It is not the case that* $\square_{\alpha}^{M} \bot$
4. $\square_{\alpha}^{M} \neg A$ *iff it is not the case that* $\square_{\alpha}^{M} A$.
5. $\square_{\alpha}^{M} A \wedge B$ *iff both* $\square_{\alpha}^{M} A$ *and* $\square_{\alpha}^{M} B$.
6. $\square_{\alpha}^{M} A \vee B$ *iff either* $\square_{\alpha}^{M} A$ *or* $\square_{\alpha}^{M} B$, *or both.*
7. $\square_{\alpha}^{M} A \rightarrow B$ *iff if* $\square_{\alpha}^{M} A$ *then* $\square_{\alpha}^{M} B$.
8. $\square_{\alpha}^{M} A \leftrightarrow B$ *iff.* $\square_{\alpha}^{M} A$ *if and only if* $\square_{\beta}^{M} B$.
9. $\square_{\alpha}^{M} A$ *iff for every β in M such that αRβ,* $\square_{\beta}^{M} A$.
10. $\square_{\alpha}^{M} \Diamond A$ *iff for some β in M such that αRβ,* $\square_{\beta}^{M} A$.

If a model M=<W,R,V> exists such that for α ∈ W we have $\square_{\alpha}^{M}$ A then we say that M satisfies A. Formally:

Definition 21

Let M=<W,R,V> be a model and A any formula. We say that M ***satisfies*** *A if there is a world* $\alpha \in W$ *such that* $\square_{\alpha}^{M} A$ *. A is said to be* ***unsatisfiable*** *if there isn't any model that satisfies the formula A (Chellas 1980:36).*

We will say that A is true at a model M if and only if it is the case that A is true at *every* world in the model M. This statement can be formalised into a definition.

Definition 22

A is true at a model M *written* $\square^{M} A$, *iff for every world* α *in M,* $\square_{\alpha}^{M} A$ *(Chellas 1980:36).*

In propositional modal logic we can group models into classes of models. The definitions given so far do not define the truth of A in a class of models and therefore we require a further definition.

Definition 23

A is ***valid in a class C of models***, *written* $\square_{C} A$, *iff for every model M in C,* $\square^{M} A$ *(Chellas 1980:36).*

A number of formulas or sets of formulas have the interesting property that they are true in all classes of standard models. The first of these is the set of *modal tautologies*.

If the modal language was to be treated as a propositional language (as defined in chapter 2) with atoms P, Q, ... □P, □Q, ...□□P, ..., ◊P, ◊Q, ... then formulas such as □P ∨ ¬□P would be tautologies in terms of definition 4 in the propositional context. In the modal context, these formulas with their tautological structure, referred to as modal tautologies, are all valid in any class of standard models. For the remainder of this dissertation the set of tautologies will be the set of all propositional and modal tautologies.

Theorem 14

If A is a tautology, then $\square_{C} A$ *for any class C of standard models.*

(Refer to Chellas 1980:37 for a proof.)

Another set of formulas that are true in all classes of standard models are the formulas that have the form ◊A → ¬□¬A, denoted by Df◊. Note that all formulas that have the same structure as Df◊ are said to be an *instance* of the schema (i.e. every formula of this form) Df◊. For instance the formulas ◊P → ¬□¬P and ◊(P → Q) → ¬□¬(P → Q) are instances of the schema Df◊.

Theorem 15

Let C be a class of standard models. Then every instance of the schema Df ◊. ◊A ↔ ¬□¬A, is valid in C, i.e.$\Box_C$ *◊A ↔ ¬ ¬A.*

(Refer to Chellas 1980:69 for details of the proof.)

Although there are other schemas that are valid in all classes of standard models we will not cover them here as we do not require their results in the ensuing sections. The interested reader is referred to the work of Chellas (1980).

From theorem 1 the reader will remember that we noted that for a given set of formulas Γ = { A_1, A_2, ..., A_n} and a formula A, Γ logically entails A if and only if $(A_1 \wedge A_2 \wedge \ldots \wedge A_n) \rightarrow A$ is valid. Modal logic on the other hand distinguishes between two types of logical entailment, namely local and global entailment. We look at local entailment first:

Definition 24

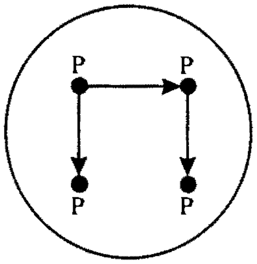
Let C be a class of models, Γ a set of formulas and A a formula. A is ***locally logically entailed*** *by Γ, with respect to C, if and only if for every M=<W,R,V> of C and for every α ∈ W, if* $\Box_{\alpha}^{M} A_i$ *for every* A_i *∈ Γ then* $\Box_{\alpha}^{M} A$.

Global entailment, on the other hand, is defined as follows:

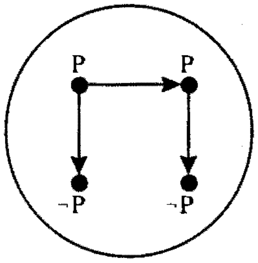
Definition 25

Let C be a class of models, Γ a set of formulas and A a formula. A is ***globally logically entailed*** *by Γ, with respect to C, if and only if for every M in C, if* $\Box^{M} A_i$ *for every* A_i *∈ Γ then* $\Box^{M} A$.

For example, consider the two formulas P and □P. P globally logically entails □P but P does not locally logically entail □P. This is because when determining if P globally logically entails □P we consider every model M ∈ C in which P is true at every world in the model M. It follows therefore by definition of the □ that □P would be true at every world at which P is in the respective models M. This is illustrated by the circle on the left below. For local logically entailment all models M ∈ C such that P is true at one or more worlds in M are considered. It follows therefore that there may be a model M such that P is true at a world w in M but □P is not true at this world, as illustrated by the circle on the right in the following diagram.



P globally logically entails □P



P does not locally logically entail □P

The following theorem holds for both local and global entailment.

Theorem 16

If the formulas A_1, ..., A_n ($n \geq 0$) are all valid in a class C of models and A is logically entailed by A_1, ..., A_n, then A is also valid in C.

Proof: We show local logical entailment first . Suppose A_1, ..., A_n ($n \geq 0$) are all valid in a class C of models and that A is a logically entailed by A_1, ..., A_n. We have to show that A is valid in C.

Choose any model M=<W, R, V> in C, and any world $\alpha \in W$. By assumption, we have that A_1, ..., A_n are true in M and therefore A_1, ..., A_n are all true at α in M, i.e. we have $\square_\alpha^M A_1 \wedge A_2 \ldots \wedge A_n$. Since A_1, ..., A_n logically entail A we know from definition 24 above that if $\square_\alpha^M A_1 \wedge A_2 \ldots \wedge A_n$ then $\square_\alpha^M A$. Since M was an arbitrarily chosen model in C, and α an arbitrary chose world in M we have that A is true at every world α in every model M of C, i.e. $\square_C A$.

For global logical entailment, suppose that A_1, ..., A_n ($n \geq 0$) are all valid in a class C of models and that A is a globally logically entailed by A_1, ..., A_n. We have to show that A is valid in C.

Choose any model M=<W, R, V> in C such that $\square^M A_1 \wedge A_2 \ldots \wedge A_n$. Since A_1, ..., A_n globally logically entail A we know from definition 25 above that if $\square^M A_1 \wedge A_2 \ldots \wedge A_n$ then $\square^M A$. Since M is an arbitrary model in C, we have that A is true in every model M of C, i.e. $\square_C A$. □

Note that for both local and global logical entailment we have that if $\square_C A_1 \wedge A_2 \wedge \ldots A_n \rightarrow A$ then $\Gamma \square_C A$, for any class of models C, formula A and set of formulas $\Gamma = \{A_1, A_2, \ldots A_n\}$. The converse, however, does not hold for both global and local entailment. Since, unlike local entailment, if Γ globally logically entails A then it need not be the case that $\square_C A_1 \wedge A_2 \wedge \ldots A_n \rightarrow A$. For instance, from the example above it should be clear that P globally logically entails □P but it is not the case that $\square_C P \rightarrow \square P$.

The following result concludes this section. This result is also applicable to all classes of standard models and is used in later sections.

Theorem 17

For $n \geq 0$, and for any class C of standard models if $\Box_C(A_1 \wedge A_2 \ldots .\wedge A_n) \rightarrow A$, then $\Box_C(A_1 \wedge \ldots .\wedge A_n) \rightarrow A$.

(Refer to Chellas 1980:69-70 for a proof.)

3.3 The Schemas L,T, B, 4 And 5

For the remainder of this chapter the schemas L, T, B, 4 and 5 will be considered for special attention. The focus is essentially on these schemas because of their historical prominence and because these schemas are important in defining different modal logics.

These schemas are defined as follows:

B. $A \rightarrow \Box\Diamond A$.

L. $\Box((A \wedge \Box A) \rightarrow B) \vee \Box((B \wedge \Box B) \rightarrow A)$

T. $\Box A \rightarrow A$.

4. $\Box A \rightarrow \Box\Box A$.

5. $\Diamond A \rightarrow \Box\Diamond A$.

Initially, it is important to note that these schemas are not valid in all classes of standard models and they are not necessarily valid or invalid in the same classes of standard models. In other words there exists classes of standard models in which each of these schemas are invalid. This result is noted in the following theorem the details which can be found in Chellas (1980:76).

Theorem 18

None of the schema's B, L, T, 4 and 5 are valid in the class of all standard models.

Despite the fact that none of the schemas D, T, B, 4 and 5 are valid in the class of all standard models they are, however, valid in specific classes of standard models. These classes have specific properties associated to the properties of the relations within their models. We will define the properties of the relations as follows: Let M = <W, R, V> be a standard model. The relation R is:

- *reflexive* iff for every $\alpha \in W$, $\alpha R \alpha$;
- *symmetric* iff for every α and $\beta \in W$, if $\alpha R \beta$, then $\beta R \alpha$;
- *transitive* iff for every α, β, and $\gamma \in W$, if $\alpha R \beta$ and $\beta R \gamma$, then $\alpha R \gamma$;
- *weakly-connected* iff for every $\alpha, \beta, \gamma \in W$, if $\alpha R \beta$ and $\alpha R \gamma$ then $\beta R \gamma$ or $\beta = \gamma$ or $\gamma R \beta$.

The model M itself is called reflexive, symmetric, transitive or weakly-connected if the relation R has these properties. If all the reflexive models were to be grouped into a class of standard models and the same was to be done for all the symmetric, transitive and weakly-connected models the following result would be apparent with regards to the schemas defined above.

Theorem 19

Let C be a class of standard models.

1. *If all the models in C are reflexive, then T is valid in C.*
2. *If all the models in C are symmetric, then B is valid in C.*
3. *If all the models in C are transitive, then 4 is valid in C.*
4. *If all the models in C are weakly-connected, then L is valid in C.*

(Refer to Chellas 1980:80 for a proof.)

4. NORMAL SYSTEMS OF MODAL LOGIC

4.1 Introduction

A system of modal logic is any set of formulas containing all the tautologies and that is closed under the rule of inference RPL, which states that from $A_1, ..., A_n$ $(n \geq 0)$ we can conclude A where A is logically entailed (i.e. logical entailment for propositional logic) by $A_1, ..., A_n$ Formally a normal system of modal logic is defined as follows:

Definition 26

A system of modal logic is defined to be normal if and only if it contains all instances of the schema Df◊ and is closed under the rule of inference $(A_1 \wedge ... \wedge A_n) \rightarrow A \,/\, (\Box A_1 \wedge .. .\wedge \Box A_n) \rightarrow \Box A$ $(n \geq 0)$ (read: from $(A_1 \wedge ... \wedge A_n) \rightarrow A$ we can infer $(\Box A_1 \wedge .. \wedge \Box A_n) \rightarrow \Box A$ $(n \geq 0)$) called RK (Chellas 1980:114).

We will call the smallest normal system of modal logic K. K contains all instances of the schema Df◊ and is closed under the rule of inference RK. Since K is the smallest normal system of modal logic it follows that all normal systems of modal logic will contain K. To simplify naming normal systems we write $KS_1...S_n$ to denote the normal system obtained by taking the schemas $S_1, ..., S_n$ as theorems. So for example KT4 is the smallest normal system produced by treating the schemas T and 4 (as defined previously) as theorems. By the notion of a theorem we mean those formulas that are contained in a system. Generally we will write $\Box_\Sigma A$ to mean that A is a theorem of a system Σ. This statement can be formalised in a definition.

Definition 27

$\Box_{\Sigma}A$ iff $A \in \Sigma$ where Σ is a normal system of modal logic.

4.2 The Schemas L, T, B, 4 And 5

In this section we will discuss the normal extensions of K obtained by adding as theorems the following schemas:

L. $\Box(A \wedge \Box A \rightarrow B) \rightarrow \Box(B \wedge \Box B \rightarrow A)$

T. $\Box A \rightarrow A$

B. $A \rightarrow \Box\Diamond A$

4. $\Box A \rightarrow \Box\Box A$

5. $\Diamond A \rightarrow \Box\Diamond A$

Including K itself there are just 15 distinct normal systems produced by taking these schemas as theorems in all possible combinations. We will, however, only look at three combinations which have received a lot of attention in the realms of artificial intelligence. These are:

1. **KT4**, the Lewis system S4.

2. **KT4L** which was first studied by Dumment & Lemmon in 1959. This system is often referred to as S4.3 and we will define it in terms of the schemas K, T, 4 and L. The reader should be aware that other authors may define it differently, for instance, Goré (1995) defines it as K, T, 4 and 3, where the schema 3 is defined as $\Box(\Box A \rightarrow B) \vee \Box(\Box B \rightarrow A)$.

3. **KT5**, the Lewis system S5.

Although KT5 is defined in terms of the schemas K, T and 5 it can also be shown that KT5 contains the schemas B and 4, that is, all instances of the schema B and all instances of schema 4 are theorems of KT5. This result will be used extensively in the following sections and it is formally proved here.

Theorem 20

The schemas B and 4 are theorems of KT5.

Proof: Before showing that KT5 contains the theorems B and 4 we require the following lemma, the proof of which can be found in Chellas (1980:114-117).

Lemma: *Every normal system of modal logic has the following rules of inference and theorems.*

RM. $A \rightarrow B \,/\, \Box A \rightarrow \Box B$

RE. $A \leftrightarrow B \,/\, \Box A \leftrightarrow \Box B$

We show that all instances of the schema B are theorems of KT5 by means of the following proof where the right hand side provides the justification for the steps on the left.

1.	$\Box\neg A \rightarrow \neg A$	schema T
2.	$A \rightarrow \neg\Box\neg A$	1, RPL
3.	$\Diamond A \leftrightarrow \neg\Box\neg A$	Df$\Diamond$
4.	$A \rightarrow \Diamond A$	2, 3, RPL
5.	$\Diamond A \rightarrow \Box\Diamond A$	schema 5.
6.	$A \rightarrow \Box\Diamond A$	4, 5, RPL

We show that all instances of the schema 4 are theorems of KT5. In order to do so we first need to show that $\Diamond\Box A \rightarrow \Box A$ and $\Diamond\Box A \leftrightarrow \neg\Box\Diamond\neg A$ are theorems of KT5. First we show that $\Diamond\Box A \leftrightarrow \neg\Box\Diamond\neg A$ is a theorem of KT5.

1.	$\Box A \leftrightarrow \neg\Diamond\neg A$	Df$\Box$ (follows by definition)
2.	$\neg\Box A \leftrightarrow \Diamond\neg A$	1,RPL
3.	$\Box\neg\Box A \leftrightarrow \Box\Diamond\neg A$	2,RE
4.	$\neg\Box\neg\Box A \leftrightarrow \neg\Box\Diamond\neg A$	3,RPL
5.	$\Diamond\Box A \leftrightarrow \neg\Box\neg\Box A$	Df$\Diamond$
6.	$\Diamond\Box A \leftrightarrow \neg\Box\Diamond\neg A$	4,5,RPL

Next we show that $\Diamond\Box A \rightarrow \Box A$:

1.	$\Diamond\neg A \rightarrow \Box\Diamond\neg A$	5
2.	$\neg\Box\Diamond\neg A \rightarrow \neg\Diamond\neg A$	1, RPL
3.	$\Box A \leftrightarrow \neg\Diamond\neg A$	Df$\Box$
4.	$\neg\Box\Diamond\neg A \rightarrow \Box A$	2,3, RPL
5.	$\Diamond\Box A \leftrightarrow \neg\Box\Diamond\neg A$	proved above
6.	$\Diamond\Box A \rightarrow \Box A$	4,5,RPL

Now we can show that 4 is a theorem of KT5:

1.	$\Diamond\Box A \rightarrow \Box A$	proved above
2.	$\Box\Diamond\Box A \rightarrow \Box\Box A$	1,RM
3.	$\Box A \rightarrow \Box\Diamond\Box A$	B
4.	$\Box A \rightarrow \Box\Box A$	2,3,RPL

It follows therefore that the KT5 system contains all instances of B and 4. □

5. SOUNDNESS

From section 4 in chapter two the reader will recall that the notion of soundness was defined. In this section we would like to explore soundness further in terms of the normal systems of modal logic. Essentially what we will show is that for each of the normal systems of modal logic defined above there exists classes of standard models in which all of the theorems contained in the normal system of modal logic are valid. For instance, if the normal system of modal logic is Σ and the class of standard models is C then every theorem in Σ is valid in C.

Definition 28

A system of modal logic Σ is ***sound*** *with respect to a class of standard models C iff every theorem of Σ is valid in C, i.e. if $\Box_{\Sigma}A$ then $\Box_{C}A$ (Chellas 1980:59).*

The following theorem provides the basis for proofs of soundness for normal modal logics with respect to classes of standard models.

Theorem 21

Let S_1, ..., S_n be schemas valid respectively in classes of standard models C_1, ..., C_n. Then the system of modal logic $KS_1...S_n$ is sound with respect to the class $C_1 \cap ... \cap C_n$

Proof: Let S_1, ..., S_n be the schemas valid in the classes of standard models C_1, ..., C_n respectively. We want to show that $KS_1...S_n$ is sound with respect to the class $C_1 \cap ... \cap C_n$.

Firstly we show that all tautologies are valid in $C_1 \cap ... \cap C_n$. This follows from theorem 14. Next we show that the class $C_1 \cap ... \cap C_n$ validates Df◇. This follows from theorem 15. Finally, we show that validity in this class is preserved by the rules of inference RK and RPL. Now RK follows from theorem 17 and RPL follows from theorem 16. This completes the proof. □

Soundness for the normal systems of modal logic KT4, KT4L and KT5 follows from the following theorem.

Theorem 22

The normal systems KT4, KT4L and KT5 are sound with respect to the classes of standard models as shown in the table below:

Normal System	***Standard Model***
KT4	*reflexive and transitive*
KT4L	*reflexive, transitive and weakly-connected*
KT5	*reflexive, symmetric and transitive*

Proof: K is sound with respect to every class of standard models, including the whole class, since this system is axiomatised (see section 8.1) by Df◇, RK and RPL.

By theorem 19 (1, 3) the schemas T and 4 are valid respectively in classes of reflexive and transitive standard models. So it follows by theorem 21 that the system of modal logic KT4 is sound with respect to the class of reflexive-transitive standard models.

By theorem 19 (1, 3, 4) the schemas T, 4 and L are valid respectively in classes of reflexive, transitive and weakly-connected standard models. Consequently it follows by theorem 21 that the system of modal logic KT4L is sound with respect to the class of weakly-connected, reflexive-transitive standard models. Note that we sometimes refer to the class of weakly-connected, reflexive-transitive standard models as the class of totally connected models.

In theorem 20 it was shown that B and 4 are theorems of KT5. By theorem 19 (1, 2, 3) the schemas T, B and 4 are valid respectively in classes of reflexive, symmetric and transitive standard models. So it follows by theorem 21 that the system of modal logic KT5 is sound with respect to the class $C_R \cap C_S \cap C_T$ of standard models, where C_R is the class of reflexive standard models, C_S is the class of symmetric standard models and C_T is the class of transitive standard models. □

6. COMPLETENESS

We stated in the previous section that a system of modal logic Σ, is sound with respect to a class of models C, if and only if every theorem of Σ is valid in C. Completeness on the other hand defines the converse relationship, that is, a system of modal logic Σ is complete with respect to a class of models C iff every formula A, that is valid in C, is a theorem of the normal system of modal logic, that is, if $\square_C A$ then $\square_\Sigma A$. We will prove completeness by using canonical standard models i.e. standard models whose worlds verify just those formulas they contain. In order to do so we need to define the concepts of consistency and maximality.

6.1 Consistency

In order to define the notion of consistency we need to understand what is meant by the term *deducible.*

Definition 29

Let Σ be a system of modal logic, Γ a set of formulas, and A any formula. A is ***deducible*** *from Γ, denoted $\Gamma \square_\Sigma A$ iff there are $A_1, \ldots, A_n \in \Gamma$ $(n \geq 0)$ such that $\square_\Sigma (A_1 \wedge \ldots \wedge A_n) \to A$ (Chellas 1980:47).*

A set of formulas Γ is said to be consistent in Σ (denoted $Con_\Sigma \Gamma$) iff the formula ⊥ is not Σ-deducible from Γ, i.e. Γ is inconsistent in Σ (denoted $C\phi n_\Sigma \Gamma$) iff $\Gamma \square_\Sigma \bot$.

Definition 30

Let Σ be a system of modal logic and Γ a set of formulas. ***$Con_\Sigma \Gamma$*** *iff it is not the case that $\Gamma \square_\Sigma \bot$ (Chellas 1980:47).*

6.2 Maximality

Intuitively, a set is maximal if it is consistent and contains as many formulas as possible without becoming inconsistent.

Definition 31

Let Σ be a system of modal logic and Γ a set of formulas. Γ is ***maximal*** *with respect to Σ, written $Max_{\Sigma}\Gamma$, if and only if:*

1. $Con_{\Sigma}\Gamma$ *and*
2. *for every A, if $Con_{\Sigma}(\Gamma \cup \{A\})$ then $A \in \Gamma$.*

If $Max_{\Sigma}\Gamma$, we say that Γ is a Σ-maximal set of formulas (Chellas 1980:47).

Before we list the properties of maximal sets of formulas we need to define a *Σ-system*. Since systems are simply sets of formulas their relative strengths can be measured in terms of inclusion, that is, a system is at least as strong as a system Σ iff it contains every theorem of Σ. If a system contains every theorem of Σ we refer to it as a Σ-system.

Theorem 23

Let Γ be a Σ-maximal set of formulas, Then:

1. $A \in \Gamma$ *iff* $\Gamma_{\Box\Sigma}A$.
2. $\Sigma \subseteq \Gamma$.
3. $T \in \Gamma$.
4. $\perp \notin \Gamma$.
5. $\neg A \in \Gamma$ *iff* $A \notin \Gamma$
6. $A \wedge B \in \Gamma$ *iff both* $A \in \Gamma$ *and* $B \in \Gamma$
7. $A \vee B \in \Gamma$ *iff either* $A \in \Gamma$ *or* $B \in \Gamma$ *or both*
8. $A \rightarrow B \in \Gamma$ *iff* $A \notin \Gamma$ *or* $B \in \Gamma$ *or both*
9. $A \leftrightarrow B \in \Gamma$ *iff both* $A \in \Gamma$ *and* $B \in \Gamma$ *or both* $A \notin \Gamma$ *and* $B \notin \Gamma$
10. Γ *is a Σ-system.*
11. $\Box A \in \Gamma$ *iff for every Σ-maximal set Δ such that* $\{A \mid \Box A \in \Gamma\} \subseteq \Delta$, $A \in \Delta$.
12. $\Diamond A \in \Gamma$ *iff for every Σ-maximal set Δ such that* $\{\Diamond A \mid A \in \Delta\} \subseteq \Gamma$, $A \in \Delta$.

(See Chellas 1980:53,158 for proof.)

6.3 Lindenbaum's Lemma

Due to the properties of maximal sets it would be convenient if we could determine whether every consistent set of formulas can be extended to a maximal set. Lindenbaum's lemma shows that this is in fact possible.

Lindenbaum's Lemma

Let Σ be a system of modal logic and Γ a set of formulas. If $Con_\Sigma\Gamma$, then there exists a Δ such that $\Gamma \subseteq \Delta$ and $Max_\Sigma\Delta$.

(Refer to Chellas 1980:55 for details of proof.)

From Lindenbaum's lemma it follows that a formula is a theorem of a normal system of modal logic Σ if and only if it is a member of every maximal set of formulas of Σ.

Theorem 24

$\vdash_\Sigma A$ iff $A \in \Delta$, for every Δ such that $Max_\Sigma\Delta$.

(Refer to Chellas 1980:57 for a proof.)

6.4 Canonical Standard Models

We need one more concept before we can present a definition of canonical standard models. This is the concept of proof sets.

Definition 32

*The **proof set** of a formula A relative to a system Σ denoted $|A|_\Sigma$ is the set of Σ-maximal sets of formulas containing A, i.e. $|A|_\Sigma = \{ \Gamma \mid Max_\Sigma\Gamma$ and $A \in \Gamma\}$(Chellas 1980:60).*

Definition 33

*Let $M = <W, R, V>$ be a standard model, and let Σ be a normal system of modal logic. M is a **canonical standard model** for Σ iff:*

1. *$W=\{\Gamma \mid Max_\Sigma\Gamma\}$.*
2. *For every α in M, $\Box A \in \alpha$ iff for every β in M such that $\alpha R\beta$, $A \in \beta$.*
3. *$V(P) = |P|_\Sigma$ for each atom P. (Chellas 1980:171.)*

Note that it is not immediately obvious from this definition that there are any such models; that is, that there are relations R satisfying the condition in clause (2) of the definition. We defer to section 6.5 the proof that such models do indeed exist. It is however, clear from this definition, that the worlds in canonical standard models verify exactly those formulas which they contain. We prove this result in the following theorem.

Theorem 25

Let M=<W,R,V> be a canonical standard model for a normal system Σ Then for every $\alpha \in W$, $\square^{M}_{\alpha} A$ iff $A \in \alpha$

Proof: Let Σ be a normal system of modal logic and let M=<W, R, V> be a canonical standard model for Σ. The proof is by induction on the complexity of A. For the inductive cases we make the hypothesis that the result holds for all formulas shorter than A. We will only consider those cases where:

- A is atomic
- A has the form $\neg$B
- A has the form B $\vee$ C
- A has the form $\square$B

If A is atomic, the result follows immediately from definition 33 (3).

Suppose A has the form $\neg$B. Choose any α in M. First, suppose that $\square^{M}_{\alpha} \neg B$. Then it is not the case that $\square^{M}_{\alpha}$ B and by the inductive hypothesis, B $\notin \alpha$. From theorem 23 (5) it follows that $\neg$B $\in \alpha$. Conversely assume $\neg$B $\in \alpha$. It follows that B $\notin \alpha$, since $\alpha \in W = \{\Gamma \mid Max_{\Sigma}\Gamma\}$. By the induction hypothesis it follows that it is not the case that $\square^{M}_{\alpha}$ B, i.e. $\square^{M}_{\alpha} \neg B$ by definition 20 (4).

Next, suppose that A has the form B $\vee$ C, and choose any $\alpha \in$ W. First, deem that $\square^{M}_{\alpha}$ B $\vee$ C, i.e. $\square^{M}_{\alpha}$ B, or $\square^{M}_{\alpha}$ C, or both. If $\square^{M}_{\alpha}$ B then by the inductive hypothesis, B $\in \alpha$ and from theorem 23 (7), B $\vee$ C $\in \alpha$. Similarly if $\square^{M}_{\alpha}$ C then B $\vee$ C $\in \alpha$. Conversely assume B $\vee$ C $\in \alpha$. Then by theorem 23 (7) B $\in \alpha$, or C $\in \alpha$, or both. From the inductive hypothesis it follows that $\square^{M}_{\alpha}$ B, or $\square^{M}_{\alpha}$ C, or both. But then by definition 20 (6) we have $\square^{M}_{\alpha}$ B $\vee$ C.

Finally, presume that A has the form $\Box$B, and pick any $\alpha \in$ W. First suppose that $\square^{M}_{\alpha}\Box$B, i.e. for every $\beta \in$ W such that $\alpha R\beta$, $\square^{M}_{\beta}$B. By the inductive hypothesis B $\in \beta$ for every β such that $\alpha R\beta$. Therefore from definition 33 (2) we have that $\Box$B $\in \alpha$. Conversely assume that $\Box$B $\in \alpha$. Then by definition 33 (2), for every $\beta \in$ W such that $\alpha R\beta$, B $\in \beta$. But then by the inductive hypothesis $\square^{M}_{\beta}$B for every $\beta \in$ W such that $\alpha R\beta$. By definition 20 (9) this is just $\square^{M}_{\alpha}\Box$B.$\square$

From the theorem above it follows that the theorems of the normal system of modal logic are only those formulas that are true in any of its corresponding canonical standard models. In other words if M=<W,R,V> is a canonical standard model for a normal system of modal logic Σ then for any well formed formula A, A is true in M if and only if A is a theorem of Σ. We prove this result below.

Theorem 26

Let M=<W, R, V> be a canonical standard model for a normal system Σ Then $\square^{M}A$ iff $\square_{\Sigma}A$.

Proof: Suppose $\square^{M}$A. Then by definition, $\square^{M}_{\alpha}$A for every $\alpha \in$ W. So by theorem 25, A $\in \alpha$ for every $\alpha \in$ W. By the definition of W, A $\in \Delta$, for every Δ such that Max$_{\Sigma}\Delta$. Therefore, by theorem 24, $\square_{\Sigma}$A. Conversely suppose $\square_{\Sigma}$A. By theorem 24, A $\in \Delta$ for every Max$_{\Sigma}\Delta$. Thus A $\in \alpha$ for every $\alpha \in$ W since W=$\{\Delta \mid \text{Max}_{\Sigma}\Delta\}$. Then by theorem 25, $\square^{M}_{\alpha}$A for every $\alpha \in$ W, i.e. $\square^{M}$A. $\square$

6.5 Proper Canonical Standard Models

As we mentioned in the previous section we do not as yet know whether a canonical model exists for a normal system of modal logic; merely what one looks like. In order to show completeness for the respective normal systems of modal logic, it is imperative that such models exist. Fortunately, we can construct models, known as proper canonical standard models, which can be shown to exist for all normal systems of modal logic. These constructed models have the advantage that they are canonical standard models and as a result we can use these proper canonical standard models to show completeness for the normal systems of modal logic. Proper canonical standard models are defined as follows:

Definition 34

Let M = <W, R, V> be a standard model, and let Σ be a normal system of modal logic. M is the ***proper canonical standard model*** *for Σ iff:*

1. *$W=\{\Gamma \mid Max_{\Sigma}\Gamma\}$*
2. *For every α and β in M, $\alpha R\beta$ iff $\{A: \Box A \in \alpha\} \subseteq \beta$*
3. *$V(P) = |P|_{\Sigma}$ for each atom P.* (Chellas 1980:173.)

Clearly models in which R is defined as in the definition above, always exist. It follows therefore that if we can show that these models are canonical standard models then we have the result that canonical models exist for normal systems of modal logic. This result in turn ensures the completeness of normal systems of modal logic.

Theorem 27

Proper canonical standard models are canonical standard models.

Proof: Let M be the proper canonical standard model for a normal system Σ. We need only to show that for every world (Σ-maximal set of formulas) $\alpha \in W$, $\Box A \in \alpha$ iff for every $\beta \in W$ (i.e. $\text{Max}_{\Sigma}\beta$) such that $\{A \mid \Box A \in \alpha\} \subseteq \beta$, $A \in \beta$. But this follows from theorem 23 (11).□

In general, to prove the completeness of a normal systems of modal logic with respect to a class of models it is sufficient to show that the proper canonical standard model for the system is contained in the class of models. Therefore, in order to show the completeness of KT4, KT4L and KT5 with respect to their associated classes of standard models, we need to show that the proper canonical models of these systems are contained in their respective classes of standard models. This result is shown in terms of the following theorem.

Theorem 28

Let M be the proper canonical standard model for a normal system Σ Then:

1. *M is reflexive if Σ contains T.*
2. *M is symmetric if Σ contains B.*
3. *M is transitive if Σ contains 4.*
4. *M is weakly-connected if Σ contains KT4L.*

Proof: (See Chellas 1980:175 for the proofs of 1-3.)

We prove 4. Let M=<W,R,V> be the proper canonical standard model for the normal system of modal logic Σ. Assume that Σ contains KT4L. Choose any α, β, $\gamma \in W$ such that $\alpha R\beta$ and $\alpha R\gamma$. Assume that R is not weakly connected, that is, it is not the case that $\beta R\gamma$ nor $\gamma R\beta$ and $\beta\neq\gamma$. Then, by the definition of a proper canonical standard model we have $\{A \mid \square A \in \beta\} \not\subseteq \gamma$. This means that there is a formula A such that $\square A \in \beta$ but $A \not\subseteq \gamma$, i.e. $\square^M_\beta \square A$ and it is not the case that $\square^M_\gamma A$. Similarly $\{B \mid \square B \in \gamma\} \not\subseteq \beta$, thus there is a formula B such that $\square B \in \gamma$ and $B \not\subseteq \beta$, i.e. $\square^M_\gamma \square B$ and it is not the case that $\square^M_\beta B$. Since Σ contains T we know that R is reflexive by 1 of this theorem and hence we also have $\square^M_\beta A$ and $\square^M_\gamma B$. From $\square^M_\beta \square A$, $\square^M_\beta A$ and not the case that $\square^M_\beta B$ we can conclude that it is not the case that $\square^M_\beta \square A \wedge A \rightarrow B$ and similarly we can conclude that it is not the case that $\square^M_\gamma \square B \wedge B \rightarrow A$. Since both β and γ are reachable from α we hold that it is not the case that $\square^M_\alpha \square(\square A \wedge A \rightarrow B)$ nor $\square^M_\alpha \square(\square B \wedge B \rightarrow A)$. By definition this means that it is not the case that $\square^M_\alpha \square(\square A \wedge A \rightarrow B) \vee \square(\square B \wedge B \rightarrow A)$, which is impossible since $\square(\square A \wedge A \rightarrow B) \vee \square(\square B \wedge B \rightarrow A)$ is an instance of L which is contained in Σ. It follows therefore from theorem 26 that $\square^M \square(A \wedge \square A \rightarrow B) \vee \square(B \wedge \square B \rightarrow A)$. Hence it follows that R must be weakly connected, that is M is weakly-connected if Σ contains KT4L.□

7. DETERMINATION

Determination is merely the combination of soundness and completeness. In other words, a normal system of modal logic Σ is said to be determined by C, a class of standard models, only when it is both sound and complete, with respect to C.

It was demonstrated in the previous two sections that the normal systems of modal logic KT4, KT4L and KT5 are both sound and complete with respect to their corresponding classes of standard models. It follows therefore that these systems are also determined by the same classes of standard models. This is formally shown by means of the following theorem.

Theorem 29

The following normal systems are determined by the classes of standard models as indicated in the table below:

Normal System	***Standard Model***
KT4	*reflexive and transitive*
KT4L	*reflexive, transitive and weakly-connected*
KT5	*reflexive, symmetric and transitive*

Proof: Soundness follows immediately from theorems 21 and 22. For completeness it is enough to observe that the proper canonical standard models for each system are in the appropriate classes of models by theorem 28.

For KT4, by parts 1 and 3 of theorem 28, the proper canonical standard model for KT4 is both a reflexive transitive standard model. Thus KT4 is determined by the class of reflexive-transitive standard models.

For KT4L, by parts 1, 3, 4 of theorem 28, the proper canonical standard model for KT4 is both reflexive, transitive and weakly-connected. Thus KT4L is determined by the class of weakly-connected reflexive-transitive standard models.

For KT5, by definition of KT5, T is a theorem of KT5. By theorem 20, B and 4 are also theorems of KT5. Therefore by theorem 28 (1, 2, 3) the proper canonical standard model for KT5 is reflexive, symmetric and transitive. Thus KT5 is determined by the class $C_R \cap C_S \cap C_T$ where C_R is the class of reflexive standard models, C_S is the class of symmetric standard models and C_T is the class of transitive standard models. □

As a result of the above theorem it is now possible to demonstrate that the three systems KT4, KT4L and KT5 are in fact all distinct.

Theorem 30

The normal systems KT4, KT4L and KT5 are all distinct.

Proof: In general, to show that a system Σ is distinct from a system Σ' it is sufficient to exhibit a model of Σ that falsifies a theorem of Σ', or vice versa. For example, to show that KT4 $\neq$ KT5, it is enough to describe a reflexive transitive standard model that falsifies an instance of the schema 5 of KT5. We follow this method in the proof below. Consider any model M=<W, R, V>, with

$W=\{\alpha, \beta\}$ $(\alpha\neq\beta)$; $R = \{<\alpha,\alpha>, <\beta,\beta>, <\alpha,\beta>\}$ $V(P)=\{\alpha\}$.

This is a reflexive transitive weakly-connected model that falsifies instances of 5 ($P \rightarrow \Box\Diamond P$ and $\Diamond P\rightarrow\Box\Diamond P$ at α). So it is a model of KT4, KT4L.

$W =\{\alpha, \beta, \gamma\}$ $(\alpha\neq\beta\neq\gamma)$; $R =\{<\alpha,\alpha>,<\beta,\beta>,<\gamma,\gamma>,<\alpha,\beta>,<\alpha,\gamma>\}$ $V(P) = \{\alpha,\beta\}$ $V(Q) = \{\alpha,\gamma\}$.

This is a reflexive transitive model of KT4 that falsifies instances of L $(\Box(P \wedge \Box P \rightarrow Q) \vee \Box(Q \wedge \Box Q \rightarrow P))$ at α.

The foregoing remarks suffice to establish the distinctness of KT4, KT4L and KT5.$\Box$

8. DECIDABILITY

Decidability was defined in chapter two as the ability of a given procedure to determine whether or not a formula A had a proof from a set of formulas Γ. This notion of decidability can also be extended to modal logic. In the realms of modal logic a decidable procedure would receive as input a set of formulas say Γ, contained in a normal system of modal logic say Σ and an arbitrary well-formed formula A. Once the information has been given to the procedure the procedure should attempt to find a proof for A from Γ. In so doing the procedure will in fact determine whether it is the case that $\Gamma\Box_C A$ where C is the corresponding class of models defined for Σ and $\Box$ is defined as in definition 24 (i.e. this procedure computes local logical entailment). If the procedure finds a proof for A from Γ, then the procedure should terminate with some sort of 'yes' answer otherwise the procedure should terminate with a 'no' answer. As with the decidable procedures in propositional logic we need to define positive ('yes') and negative ('no') tests for the modal decidable procedures.

Positive tests in modal logic can be defined in terms of axiomatisability and negative tests can be defined in terms of the finite model property. In other words, if we can show that a normal system of modal logic is axiomatisable (section 8.1) and if it has the finite model property (section 8.2) we can safely conclude that the normal system of modal logic is decidable.

8.1 Axiomatisability

We say that a rule of inference is reasonable if there is an effective way of telling when formulas are related to the rule of inference in terms of its hypothesis and conclusion. For example, the rule modus ponens is reasonable, since it is a decidable matter whether three formulas are of the forms A→B, A and B (if so, the first two are hypotheses of modus ponens and the last is a conclusion).

Now, every system of modal logic Σ can be regarded as the set of formulas generated from some subset Γ of its theorems by a set of rules of inference. This is trivial, since Σ is always generated from Σ by the rule A/A. But when Γ is a decidable set of formulas and the rules of inference are reasonable and finite in number, Σ is said to be *axiomatisable*, and Γ is said to be a set of axioms for Σ. Together the axioms and rules constitute an axiomatisation of the logic.

Axiomatisable systems are important because they admit a notion of proof and hence a positive test for theoremhood. By the notion of *a proof in an axiomatisable system* we mean a finite sequence of formulas each of which is either an axiom or follows from previous formulas in the sequence by one of the rules of inference. Axiomatisation is shown via the following theorem.

Theorem 31

Each of the normal systems KT4, KT4L and KT5 is axiomatisable by a finite number of schemas.

Proof: It is sufficient to observe that in each case the logic can be axiomatised by a finite number of schemas together with the rules of inference RPL and RK. In other wotds, each logic can be characterised by a finite set of axioms and it is a decidable exercise to determine whether any formula either matches an axiom schema or matches the structure of a given inference rule by means of its hypothesis and conclusion.

KT4 is axiomatised by means of the schemas K, T, 4, Df◊, the set of tautologies (see page 39) and the rules RPL and RK. These axioms form a decidable set and the rules are reasonable.

KT4L is axiomatised by means of the schemas K, T, 4, L, Df◊, the set of tautologies (see page 39) and the rules RPL and RK. These axioms form a decidable set and the rules are reasonable.

KT5 is axiomatised by means of the schemas K, T, 5, Df◊, the set of tautologies (see page 39) and the rules RPL and RK. These axioms form a decidable set and the rules are reasonable. □

8.2 Finite Model Property

Since axiomatisability defines a positive test for theoremhood the only outstanding requirement is the negative test for theoremhood. However, showing that normal systems of modal logic have negative tests for theoremhood relies on the finite model property (defined below) which in turn relies on the concept of filtrations. Before defining filtrations we need to ensure that we understand what we mean by a subformula of a formula A.

A *subformula* of a formula A is any formula that is a part of A, including A itself. This idea is captured in the following recursive definition of the set Sn(A) of subformulas of A (Chellas 1980:28).

Definition 35

1. $Sn(P) = \{P\}$ *for any atom* P

2. $Sn(T) = \{T\}$

3. $Sn(\bot) = \{\bot\}$

4. $Sn(\neg A) = \{\neg A\} \cup Sn(A)$

5. $Sn(A \wedge B) = \{A \wedge B\} \cup Sn(A) \cup Sn(B)$

6. $Sn(A \vee B) = \{A \vee B\} \cup Sn(A) \cup Sn(B)$

7. $Sn(A \rightarrow B) = \{A \rightarrow B\} \cup Sn(A) \cup Sn(B)$

8. $Sn(A \leftrightarrow B) = \{A \leftrightarrow B\} \cup Sn(A) \cup Sn(B)$

9. $Sn(\Box A) = \{\Box A\} \cup Sn(A)$

10. $Sn(\Diamond A) = \{\Diamond A\} \cup Sn(A)$

Note that a set of formulas is closed under subformulas if and only if the set contains every subformula of every formula it contains. Also note that if a set of formulas Γ is finite then its set of subformulas will also be finite.

We can use the concept of subformulas to define the equivalence relation ≡ as follows: Let Γ be a set of formulas closed under subformulas. For any model M = <W, R, V> we define the equivalence relation ≡ on the worlds in M by the stipulation that, for α, β ∈ W, α ≡ β iff for every A ∈ Γ, $\square_{\alpha}^{M}$ A if and only if $\square_{\beta}^{M}$ A, that is, the worlds in M are equivalent under ≡ iff they agree on every formula in Γ, the set of formulas closed under subformulas. Furthermore we define for each α in M the equivalence class [α] = {β ∈ W | α≡β}, as well as ≡ -equivalence classes of sets of worlds in M by [X] = {[α]: α ∈ X}, for every X ⊆ W. We define filtrations formally (Chellas 1980:101).

Definition 36

Let $M = <W,R,V>$ be a standard model, and let Γ be a set of formulas closed under subformulas. The ***filtration*** *of M through Γ is any standard model $M^* = <W^*, R^*, V^*>$ such that:*

(1) $W^* = [W]$

(2) *For every α and β in M:*

(a) *if $\alpha R \beta$ then $[\alpha]R^*[\beta]$;*

(b) *if $[\alpha]R^*[\beta]$, then for every formula $\Box A \in \Gamma$,*

if $\Box^{M}_{\alpha}$ A, then $\Box^{M}_{\beta} A$;

(c) *if $[\alpha]$ $R^*[\beta]$, then for every formula $\Diamond A \in \Gamma$,*

if $\Box^{M}_{\beta} A$, then $\Box^{M}_{\alpha}$ $\Diamond A$;

(3) *$V^*(P) = [V(P)]$ for each atom P in Γ ($V^*(Q)$ can be anything if Q is an atom not in Γ).*

An important aspect of a filtration M* of M through Γ is that a world α in M and its equivalence class [α] in M* agree on every formula in Γ. This result is formally stated in the following theorem.

Theorem 32

Let $M^* = <W^*, R^*, V^*>$ be a Γ-filtration of a standard model $M = <W,R,V>$. Then for every $A \in \Gamma$

$\Box^{M}_{\alpha} A$ iff $\Box^{M^*}_{[\alpha]} A$.

(See Chellas:101 for the proof.)

If M*, M and A are defined as in the theorem above it should be clear that A will be true in M if and only if A is true in M*. Formally:

Theorem 33

Let M^* be a Γ-filtration of a standard model M. Then M and M^* are equivalent modulo Γ, i.e. for every $A \in \Gamma$, $\Box^{M} A$ iff $\Box^{M^*} A$.

(See Chellas 1980:102 for proof.)

Once again if M, M* and A are defined as in the theorems above it should be clear that we can extend this result to classes of standard models, that is, A is true in every model M in a class of standard models, say C, if and only if A is true in every model M* in the class of Γ-filtrations of the models in C.

Theorem 34

Let C be a class of standard models and let Γ(C) be the class of Γ-filtrations of models in C. Then for every A ∈ Γ, $\square_{C}A$ *iff* $\square_{\Gamma(C)}A$.

It is important to observe that a filtration through a finite set of formulas always yields a finite model. For if n is the number of formulas in Γ then a filtration through Γ is a model having at most 2^n worlds, that being the maximum number of ways that worlds can agree on formulas in Γ. Note that 2^n is finite if n is.

Definition 37

Two formulas A and B are said to be ***M-equivalent*** *iff they are true at exactly the same worlds in a model M=<W,R,V>. A set of formulas Γ is* ***logically finite*** *relative to a model M if every formula in Γ is M-equivalent to one or another of a finite number of formulas in Γ (Chellas 1980:36).*

Generally, if a set of formulas Γ is logically finite relative to a model M, then every Γ-filtration of M is also finite.

From the above results it follows that if a filtration can be defined for a standard model then it would be possible to show that the normal systems of modal logic such as KT4, KT4L and KT5 are determined by classes of *finite* standard models. The advantage of this result would be that a decidable procedure would only have to consider finite models when assessing whether a formula A is a theorem or not of a normal system of modal logic.

The following two theorems provide the basis for the definition of finite standard models for the normal systems of modal logic KT4 and KT5. KT4L will be treated separately below.

Theorem 35

Let M =<W*, R*, V*> be a filtration of a standard model M=<W,R,V>. Then M* is reflexive if M is.*

Proof: Let M* =<W*, R*,V*> be a filtration of a standard model M=<W,R,V>.

Suppose that M is reflexive. Then αRα for every α in M. By definition it follows that [α]R*[α] for every [α] in M*, that is, M* is reflexive. □

There are no analogous results for arbitrary filtrations of symmetric, transitive and weakly connected models. But we can define filtrations that do have certain combinations of these properties:

Let Γ be a set of formulas closed under subformulas, and let M=<W,R,V> be a standard model. We consider the following conditions on worlds α and β in M:

C1: for every $\Box A \in \Gamma$ if $\Box^{M}_{\alpha} \Box A$ then $\Box^{M}_{\beta} A$

for every $\Diamond A \in \Gamma$ if $\Box^{M}_{\beta} A$ then $\Box^{M}_{\alpha} \Diamond A$

C2: for every $\Box A \in \Gamma$ if $\Box^{M}_{\beta} \Box A$ then $\Box^{M}_{\alpha} A$

for every $\Diamond A \in \Gamma$ if $\Box^{M}_{\alpha} A$ then $\Box^{M}_{\beta} \Diamond A$

C3: for every $\Box A \in \Gamma$ if $\Box^{M}_{\alpha} \Box A$ then $\Box^{M}_{\beta} \Box A$

for every $\Diamond A \in \Gamma$ if $\Box^{M}_{\beta} \Diamond A$ then $\Box^{M}_{\alpha} \Diamond A$

Theorem 36

Let M = <W*, R*, V*> be a standard model in which W* and V* are defined as in a Γ-filtration of a standard model M=<W,R,V>. Then:*

1. *If R* is defined by C1 and C2 then:*

(a) *M* is symmetric and*

(b) *M* is a Γ-filtration of M if M is symmetric.*

2. *If R* is defined by C1 and C3 then:*

(a) *M* is transitive, and*

(b) *M* is a Γ-filtration of M if M is transitive.*

3. *If R* is defined by C1, C2 and C3 then:*

(a) *M* is symmetric and transitive and*

(b) *M* is a Γ-filtration of M if M is symmetric and transitive.*

(See Chellas 1980:106 for the proof.)

From the above two theorems it is now possible to define the finite standard models for the systems KT4, KT5. This is done as follows:

Theorem 37

The following normal systems are determined by the corresponding classes of finite standard models as indicated in the table below.

Normal System	***Finite Standard Model***
KT4	*reflexive and transitive*
KT5	*reflexive, symmetric and transitive*

Proof: Soundness in each case is a consequence of theorem 29. The completeness of all the systems may be proved using theorems 35 and 36. We refer the interested reader to Chellas (1980:187) for the details. □

Consider for a moment a formula A that is not a theorem of some normal system say Σ which is determined by a class of finite standard models C. In order to show that A is not a theorem of Σ we will have to find at least one model M=<W,R,V> in C such that A is not true in M. In order to show that A is not true in M we would have to find at least one world say $\alpha \in W$ at which A is not true. Since M is finite, this task is finite.

Definition 38

*A modal logic Σ has the **finite model property** if and only if each non-theorem of Σ is false in some finite model of Σ (Chellas 1980:62).*

Clearly if the systems KT4, KT4L and KT5 have the finite model property it will be possible to determine if a given formula is a non-theorem of either KT4, KT4L or KT5. The following theorem establishes this result.

Theorem 38

Each of the normal systems, KT4, KT4L, KT5 have the finite model property.

Proof: Let Σ be any normal system. To prove that Σ has the finite model property we first prove that Σ is determined by a class of standard models C, that is for every A, $\Box_{\Sigma}A$ iff $\Box_{C}A$. From this result, by means of filtrations, we show that Σ is determined by the class C_{FIN} of finite standard models in C, that is, for every A, $\Box_{\Sigma}A$ iff $\Box_{C_{FIN}}$ A. Then we know that Σ has the finite model property i.e. if it is not the case that $\Box_{\Sigma}A$ then there is a finite model $M \in C_{FIN}$ such that it is not the case that $\Box^{M}A$.

We will use this approach to prove the finite model property for KT4 and KT5. By theorem 29 we know that the normal systems KT4 and KT5 are determined by the respective classes of standard models as defined in the table below:

Normal System	**Standard Model**
KT4	reflexive and transitive
KT5	reflexive, symmetric and transitive

Also by theorem 37 we know that KT4 and KT5 are determined by their corresponding classes of *finite* standard models as defined in the table above. Thus from the discussion above the result follows that KT4 and KT5 have the finite model property.

KT4L has the finite model property. Although this result is not formally proved here the reader is referred to Hughes & Cresswell (1996:157) for more details with regards to this result. □

8.3 Decidability

Recall that it was noted at the beginning of this section that a modal logic is decidable if it has both the finite model property and is axiomatisable by a finite number of schemas. In section 8.1 we showed that the normal systems KT4, KT4L and KT5 are axiomatisable by a finite number of schemas and in section 8.2 we demonstrated the finite model property for the same systems. It follows therefore that KT4, KT4L and KT5 are decidable. Formally:

Theorem 39

Each of the normal systems KT4, KT4L, KT5 are decidable.

Proof: By theorem 38 KT4, KT4L and KT5 have the finite model property. In theorem 31 we showed that KT4, KT4L and KT5 are axiomatisable by a finite number of schemas. Hence it follows that the normal systems KT4, KT4L and KT5 are decidable. □

Let Σ be a normal system of modal logic that is axiomatisable and has the finite model property. Let M_1, M_2, M_3, ... be a complete enumeration of the finite models of Σ. Let A be any formula. In order to show whether or not A is a theorem of Σ a decidable procedure would have to do the following:

1. To determine if A is a theorem of Σ the procedure would have to determine if A either matches one of the axioms schemas or the structure of the inference rules as described in theorem 31. This task is a finite exercise since Σ is axiomatised by a finitely many schemas and rules of inference.

2. To determine if A is not a theorem of Σ the procedure would have to proceed through each of the M_j ($j \geq 1$) and determine the following:

2.1 Determine whether the model M_j is a model of Σ.

2.2 If M_j is a model of Σ, then determine whether M_j falsifies A.

Consider task 2.1. By theorem 29 we know that the corresponding schemas of Σ are valid in the class of models corresponding to Σ. Therefore in order to determine whether M_j is a model of Σ, all the procedure needs to do is to determine whether the schemas that axiomatise Σ are valid in M_j. Since Σ is axiomatised by a finite number of schemas and M_j is finite, this task is finite.

Task 2.2 is also a finite exercise since the M_j's are finite models. (Chellas 1980.)

It follows that this procedure will eventually either find a model M_i for some i that falsifies A in which case A is not a theorem of Σ. Alternatively the procedure will succeed in showing that A is a theorem of Σ.

9. CONCLUSION

Decidability concludes the study of propositional modal logic including the normal systems of modal logic KT4 KT4L and KT5. The interested reader is referred to the work of modal logic language researchers such as Hughes & Cresswell (1996) and Chellas (1980) for more details with regards to the modal language and that of other normal systems of modal logic. The next chapter will provide a survey of the different resolution proof systems available to propositional modal logic, using the normal systems of modal logic presented in this chapter to demonstrate the applications of these proof systems.

CHAPTER 4

RESOLUTION WITHIN MODAL LOGIC

Science does not provide one clear and uncontested explanation. Its methods often provide a licence for different explanations at varying levels, which often are unlikely to be all equally right.

- R Trigg

1. INTRODUCTION

As mentioned in chapter 1 resolution has enjoyed much prominence in logic due to its efficient proof techniques. In this chapter we will focus on work done in resolution where its application has been that of modal logic. This chapter is by no means a complete survey of all the work done with regards to resolution within modal logic. We have however, selected a broad range of articles that we hope will at least cover the majority of fields that are currently prevalent in modal logic with regards to resolution.

Research articles that focus primarily on propositional modal resolution or alternatively provide some exposition on propositional modal resolution will be discussed fully in terms of KT4, KT4L and KT5. The following headings will be addressed for each of the selected articles.

- An overview of the authors approach.
- Notes on syntax and semantics.
- Applicable rules for the respective resolution systems (proofs that use resolution)
- Soundness and completeness results.
- Applications of the resolution system.

The reader should note that the focus of this chapter is on *how* these different resolution systems work. It is not therefore a formal exposition of these resolution systems or the theory that underlies them. The interested reader is referred to the respective research articles for the required technical details and other formalities. Also, note that for the sake of conformity with previous chapters, the notation used to describe the respective resolution systems may not correspond to that used in the original articles. For instance, this chapter may make use of '$\perp$' instead of the constant 'false'.

This chapter will be concluded with a brief comparative analysis of the different resolution systems presented here.

2. MODAL THEOREM PROVING

2.1 Overview

In this section we discuss a technique defined by Abadi & Manna (1986). Although they extend their technique to a large number of modal systems, we will only be looking at the application of this method to the propositional modal systems of KT4 and KT5.

This approach is essentially a non-clausal approach, that is, the resolution system works with formulas and converts them on the fly in an attempt to prove a contradiction. A refutation proof using this resolution system is similar to the one defined in chapter 2 (see definition 13), that is, there is a sequence of formulas $A_1, A_2, \ldots, A_n$ such that $A_1 = \neg A$, and $A_n = \bot$ and A_{i+1} is obtained from A_i by an application of a rule.

In this system finding a proof for a formula A effectively means that we have to show that $\Gamma \square_C A$ where $\square$ is defined as local logical entailment (see definition 24), Γ is the *empty* set, and C is the class of standard models corresponding to the normal systems of modal logic KT4 or KT5.

2.2 Syntax

Abadi & Manna's (1986) resolution system makes use of formulas that are defined in terms of the adequate set of connectives: $\vee$, $\wedge$ and $\neg$ and the modal connectives $\Diamond$, $\Box$. For the remainder of this section it will therefore be assumed that all the formulas that the resolution system has to work with are in this form. No conversion rules are supplied with this resolution system and it is therefore assumed that the reader may make use of the conversion rules defined in chapter 2 to eliminate the connectives $\rightarrow$, $\leftrightarrow$ etc.

2.3 Rules

When applying resolution in this context there are two types of rules that can be used:

1. Simplification rules
2. Deduction rules.

2.3.1 *Simplification Rules*

Simplification rules have the form $A_1, A_2, \ldots, A_n \Rightarrow A$, and they are treated as replacement rules, that is, if $A_1, A_2, \ldots, A_n$ occur in any order, in any conjunction, we can replace them with A. For example if we are given the rule $A, \neg A \Rightarrow \bot$ and the formula $(Q \vee \Diamond(\neg P \wedge Q \wedge P))$ we can replace $\neg P \wedge P$ with $\bot$ so that the formula becomes $(Q \vee \Diamond(Q \wedge \bot))$.

Abadi & Manna (1986) define the following simplification rules, where A and B denote formulas:

Name	Definition
true-false simplification	$\bot \vee A \Rightarrow A$ $A \vee \bot \Rightarrow A$ $\bot, A \Rightarrow \bot$ $\Diamond\bot \Rightarrow \bot$ $\neg A, A \Rightarrow \bot$
Negation rules	$\neg\Box A \Rightarrow \Diamond\neg A$ $\neg\Diamond A \Rightarrow \Box\neg A$ $\neg(A \wedge B) \Rightarrow (\neg A \vee \neg B)$ $\neg(A \vee B) \Rightarrow (\neg A \wedge \neg B)$ $\neg T \Rightarrow \bot$ $\neg\bot \Rightarrow T$
Weakening rule	$A, B \Rightarrow A$
Distribution rule	$A, B_1 \vee B_2 \vee \ldots \vee B_n \Rightarrow (A \wedge B_1) \vee (A \wedge B_2) \vee \ldots \vee (A \wedge B_n)$,

2.3.2 *Deduction Rules*

Deduction rules are of the form $A_1, A_2, \ldots A_n \mapsto A$ and are treated as addition rules, that is, if the formulas $A_1, A_2, \ldots A_n$ occur in any order, in any conjunction, then another conjunction containing A can be added. For example, if the rule $\Box A, \Diamond B \mapsto \Diamond(A \wedge B)$, where A and B are formulas, is applied to the formula $Q \vee (\Diamond Q \wedge R \wedge \Box P)$, then the formula would become $Q \vee (\Diamond Q \wedge R \wedge \Box P \wedge \Diamond(P \wedge Q))$ after the application of the rule.

Definition 39

*The scope of a well formed formula *A with * a modal connective is just A (Chan 1987:158).*

According to the resolution principle (see definition 11) for the propositional case it is possible to delete complementary (see definition 9) literals from clauses and derive a resolvent. In propositional logic these complementary literals are always within the same scope. This is unfortunately not the case with modal operators. For instance, consider the formula $P \wedge \Diamond\neg P$. Although P and $\neg P$ are complementary literals they are not within the same scope, i.e. their truth values are interpreted at different worlds and they therefore do not necessarily contradict each other. It follows therefore that if one is going to apply resolution in the modal case one is going to have to ensure that complementary literals are interpreted at the *same* world. It is because of this difficulty that Abadi & Manna (1986) introduced the following resolution rule.

The resolution rule, denoted $A\langle S, \ldots., S\rangle, B\langle S, \ldots,S\rangle \mapsto A\langle T\rangle \vee B\langle\bot\rangle$, is an instance of a deduction rule and is applied as follows: If A and B have a common sub-formula S then we can derive the resolvent $A\langle T\rangle \vee B\langle\bot\rangle$ whereby we substitute T for the one or more occurrences of S in A (i.e. only those occurrences that we wish to replace) and $\bot$ for one or more occurrences of S in B, provided that the occurrences of S in A and B that are replaced by T and $\bot$, are within the same scope. So for example, if we have the formula $\Diamond((P\vee Q) \wedge (\neg P \vee Q))$ we can apply the resolution rule to get $\Diamond((P\vee Q) \wedge (\neg P \vee Q) \wedge (\bot\vee Q) \wedge (\neg T \vee Q))$ but we cannot apply the resolution rule to $((P \vee Q) \wedge \Diamond(\neg P \vee Q))$ since $\neg P$ is within the scope of a $\Diamond$ and P is not.

The following deduction rules are applicable to the two modal systems KT4 and KT5. Note that the rules are only applicable to their corresponding modal logic systems. For any formula A or B we have:

System	**Rules**
KT4	Resolution Rule $\Box A, \Diamond B \mapsto \Diamond(\Box A \wedge B)$ $\Box A \mapsto A$
KT5	Resolution Rule $\Box A, \Diamond B \mapsto \Diamond(\Box A \wedge B)$ $\Diamond A, \Diamond B \mapsto \Diamond(\Diamond A \wedge B)$ $\Box A \mapsto A$ $A \mapsto \Diamond A$

Although Abadi and Manna (1986) list additional rules we have only extracted those that are relevant to the systems KT4 and KT5. The interested reader is referred to their article for the additional rules.

2.3.3 *Summary*

The rules for each of the systems KT4 and KT5 can be summarised by means of the following table.

System	Rules
KT4	True-false simplification
	Negation rules
	Weakening rule
	Distribution rule
	Resolution rule
	$\Box A, \Diamond B \mapsto \Diamond(\Box A \wedge B)$
	$\Box A \mapsto A$
KT5	True-false simplification
	Negation rules
	Weakening rule
	Distribution rule
	Resolution rule
	$\Box A, \Diamond B \mapsto \Diamond(\Box A \wedge B)$
	$\Diamond A, \Diamond B \mapsto \Diamond(\Diamond A \wedge B)$
	$\Box A \mapsto A$
	$A \mapsto \Diamond A$

2.4 Soundness

Theorem 40

The systems of modal logic KT4 and KT5 are sound with respect to their corresponding classes of models.

Proof: As discussed above the proof is based on the fact that the simplification and deduction rules are sound. In other words, if we start with a satisfiable formula we can only apply the simplification rules or deduction rules for KT4 (KT5). If we apply the simplification rules it follows by the soundness of the simplification rules that we will derive a new satisfiable formula. Similarly, if we apply the deduction rules (which includes the resolution rule) of KT4 (KT5) it also follows by the soundness of the deduction rules that we will derive a new satisfiable formula. (Refer to Abadi & Manna 1986:175 for details of proof.) □

2.5 Completeness

Theorem 41

The resolution systems for propositional KT4 and KT5 are complete for their corresponding classes of models.

Proof: Although no formal proof is given for KT4 and KT5 a proof sketch is provided which is based on two notions that were initially introduced by Fitting (1983). The first of these is the consistency property which 'is a syntactic property of sets of sentences that satisfies certain conditions depending on the logic' (Abadi & Manna 1986:177). The second is that of model existence which is a lemma that guarantees that 'if a set of sentences satisfies a consistency property then all the sentences in the set are satisfiable' (Abadi & Manna 1986:177). The consistency property in this instance is shown to be 'admissible' defined as:

Definition

A set of formulas Γ is ***admissible*** *for KT4 (KT5) if no finite conjunction of members of Γ can be refuted in the resolution system for KT4 (KT5).*

Admissibility can be shown to satisfy the conditions for the consistency property for KT4 and KT5 as follows:

For KT4 the admissible consistency property is defined as:

- Γ does not contain A and $\neg$A for any literal A..
- Γ does not contain $\perp$ or $\neg$T.
- if $(A \wedge B) \in \Gamma$ then $\Gamma \cup \{A, B\}$ is admissible
- if $\neg(A \vee B) \in \Gamma$ then $\Gamma \cup \{\neg A, \neg B\}$ is admissible
- if $(A \vee B) \in \Gamma$ then $\Gamma \cup \{A\}$ is admissible or $\Gamma \cup \{ B\}$ is admissible
- if $\neg(A \wedge B) \in \Gamma$ then $\Gamma \cup \{\neg A\}$ is admissible or $\Gamma \cup \{\neg B\}$ is admissible
- if $\Diamond A \in \Gamma$ then $\{\Box B \mid \Box B \in \Gamma\} \cup \{A\}$ is admissible
- if $\Box A \in \Gamma$ then $\Gamma \cup \{A\}$ is admissible

For KT5 the admissible consistency property is defined as:

- Γ does not contain A and $\neg$A for any literal A..
- Γ does not contain $\perp$ or $\neg$T.
- if $(A \wedge B) \in \Gamma$ then $\Gamma \cup \{A, B\}$ is admissible
- if $\neg(A \vee B) \in \Gamma$ then $\Gamma \cup \{\neg A, \neg B\}$ is admissible
- if $(A \vee B) \in \Gamma$ then $\Gamma \cup \{A\}$ is admissible or $\Gamma \cup \{ B\}$ is admissible
- if $\neg(A \wedge B) \in \Gamma$ then $\Gamma \cup \{\neg A\}$ is admissible or $\Gamma \cup \{\neg B\}$ is admissible
- if $\Diamond A \in \Gamma$ then $\{\Diamond B \mid \Diamond B \in \Gamma\} \cup \{\Box C \mid \Box C \in \Gamma\} \cup \{A\}$ is admissible
- if $\Box A \in \Gamma$ then $\Gamma \cup \{A\}$ is admissible

The Model Existence lemma can therefore be defined in terms of the admissible property as follows:

If a set of formulas Γ is admissible for KT4 (KT5) then Γ is satisfiable.

The proof is essentially based on a contrapositive argument, that is, we assume that there does not exist a proof for some formula A and then show that A cannot be valid in the corresponding class of models for KT4 (KT5). The reader is referred to Abadi & Manna (1986:177,8) and Fitting (1983) for the details of the proof. □

2.6 Applications

In this section we demonstrate how to use the rules in this proof. Firstly from soundness we know that if there exists a proof for a formula A using this resolution system then A is satisfied by every model M (in the appropriate class of models). So for the purpose of this example assume that we want to show that the formula $\Box(\neg(P \wedge \Box P) \vee Q) \vee \Box(\neg(Q \wedge \Box Q) \vee P)$ is satisfiable in the class of KT5 models. From the discussion above this means that we start with the negation of this formula and attempt to derive $\perp$. Note that we are working with the KT5 system and will therefore only use the deduction rules applicable to the KT5 system.

Before the proof is demonstrated, a note on the application of the rules is necessary. Both the simplification and deduction rules are based on a type of pattern matching. We attempt to find a pattern on the left hand side of the formula that matches the left hand side of the rules and the apply the right hand side of the rules changing the initial formula accordingly. For instance if we look at the rule $\neg\Box A \Rightarrow \Diamond\neg A$ and the formula $\neg\Box(P \wedge Q)$ then it should be clear that this formula is an instance of $\neg\Box A$. It follows therefore that we can apply the rule to obtain $\Diamond\neg(P \wedge Q)$.

In the proof below the formulas that are being worked on are on the left and justifications for the changes to the formulas are on the right.

1.	$\neg(\Box(\neg(P \wedge \Box P) \vee Q) \vee \Box(\neg(Q \wedge \Box Q) \vee P))$	
2.	$\neg\Box(\neg(P \wedge \Box P) \vee Q) \wedge \neg\Box(\neg(Q \wedge \Box Q) \vee P)$	negation rules
3.	$\Diamond\neg(\neg(P \wedge \Box P) \vee Q) \wedge \Diamond\neg(\neg(Q \wedge \Box Q) \vee P)$	negation rules
4.	$\Diamond(\neg\neg(P \wedge \Box P) \wedge \neg Q) \wedge \Diamond(\neg\neg(Q \wedge \Box Q) \wedge \neg P)$	negation rules
5.	$\Diamond(\neg(\neg P \vee \neg\Box P) \wedge\neg Q) \wedge \Diamond(\neg(\neg Q \vee \neg\Box Q) \wedge \neg P)$	negation rules
6.	$\Diamond(\neg\neg P \wedge \neg\neg\Box P \wedge\neg Q) \wedge \Diamond(\neg\neg Q \wedge \neg\neg\Box Q \wedge \neg P)$	negation rules
7.	$\Diamond(\neg\neg P \wedge \neg\Diamond\neg P \wedge\neg Q) \wedge \Diamond(\neg\neg Q \wedge \neg\Diamond\neg Q \wedge \neg P)$	negation rules
8.	$\Diamond(\neg\neg P \wedge \Box\neg\neg P \wedge\neg Q) \wedge \Diamond(\neg\neg Q \wedge \Box\neg\neg Q \wedge \neg P)$	negation rules
9.	$\Diamond(\Box\neg\neg P \wedge \neg Q) \wedge \Diamond(\neg\neg Q \wedge \Box\neg\neg Q \wedge \neg P)$	weakening rule
10.	$\Diamond(\neg Q) \wedge \Diamond(\neg\neg Q \wedge \Box\neg\neg Q \wedge \neg P)$	weakening rule
11.	$\Diamond(\neg Q) \wedge \Diamond(\neg\neg Q \wedge \Box\neg\neg Q)$	weakening rule
12.	$\Diamond(\neg Q) \wedge \Diamond(\Box\neg\neg Q)$	weakening rule
13.	$\Diamond(\neg Q) \wedge \Diamond(\Box\neg\neg Q) \wedge \Diamond(\Diamond(\neg Q) \wedge \Box\neg\neg Q)$	KT5 rule
14.	$\Diamond(\Box\neg\neg Q) \wedge \Diamond(\Diamond(\neg Q) \wedge \Box\neg\neg Q)$	weakening rule
15.	$\Diamond(\Diamond(\neg Q) \wedge \Box\neg\neg Q)$	weakening rule
16.	$\Diamond(\Diamond(\neg Q) \wedge \Box\neg\neg Q \wedge \Diamond(\Box\neg\neg Q \wedge \neg Q))$	KT5 rule
17.	$\Diamond(\Box\neg\neg Q \wedge \Diamond(\Box\neg\neg Q \wedge \neg Q))$	weakening rule
18.	$\Diamond(\Diamond(\Box\neg\neg Q \wedge \neg Q))$	weakening rule
19.	$\Diamond(\Diamond(\Box\neg\neg Q \wedge \neg Q \wedge \neg\neg Q))$	KT5 rule
20.	$\Diamond(\Diamond(\neg Q \wedge \neg\neg Q))$	weakening rule
21.	$\Diamond(\Diamond(\bot))$	true-false simplification
22.	$\Diamond(\bot)$	true-false simplification
23.	$\bot$	true-false simplification

Since $\bot$ was derived in the last step we can conclude by soundness that the class of models corresponding to KT5, satisfies the formula $\Box(\neg(P \wedge \Box P) \vee Q) \vee \Box(\neg(Q \wedge \Box Q) \vee P)$.

To demonstrate the use of the resolution rule we show that the formula $\neg\Box P \vee \Box\Box P$ is satisfied by the class of models corresponding to KT4. Negating the formula the proof follows:

1.	$\neg(\neg\Box P \vee \Box\Box P)$	
2.	$\neg(\Diamond\neg P \vee \Box\Box P)$	negation rules
3.	$\neg\Diamond\neg P \wedge \neg\Box\Box P$	negation rules
4.	$\Box\neg\neg P \wedge \neg\Box\Box P$	negation rules
5.	$\Box\neg\neg P \wedge \Diamond\neg\Box P$	negation rules
6.	$\Box\neg\neg P \wedge \Diamond\Diamond\neg P$	negation rules
7.	$\Box\neg\neg P \wedge \Diamond\Diamond\neg P \wedge \Diamond(\Box\neg\neg P \wedge \Diamond\neg P)$	KT4 rule
8.	$\Diamond\Diamond\neg P \wedge \Diamond(\Box\neg\neg P \wedge \Diamond\neg P)$	weakening rule
9.	$\Diamond(\Box\neg\neg P \wedge \Diamond\neg P)$	weakening rule
10.	$\Diamond(\Box\neg\neg P \wedge \Diamond\neg P \wedge \Diamond(\Box\neg\neg P \wedge \neg P))$	KT4 rule
11.	$\Diamond(\Diamond\neg P \wedge \Diamond(\Box\neg\neg P \wedge \neg P))$	weakening rule
12.	$\Diamond(\Diamond(\Box\neg\neg P \wedge \neg P))$	weakening rule
13.	$\Diamond(\Diamond(\Box\neg\neg P \wedge \neg P \wedge \neg\neg P))$	KT4 rule
14.	$\Diamond(\Diamond(\Box\neg\neg P \wedge \neg P \wedge \neg\neg P)) \wedge \Diamond(\Diamond(\Box\neg\neg P \wedge \neg T \wedge \neg\neg\bot))$	resolution rule
15.	$\Diamond(\Diamond(\Box\neg\neg P \wedge \neg T \wedge \neg\neg\bot))$	weakening rule
16.	$\Diamond(\Diamond(\Box\neg\neg P \wedge \bot \wedge \neg\neg\bot))$	negation rule
17.	$\Diamond(\Diamond(\Box\neg\neg P \wedge \bot))$	weakening rule
18.	$\Diamond(\Diamond(\bot))$	weakening rule
19.	$\Diamond(\bot)$	true-false simplification
20.	$\bot$	true-false simplification

It follows therefore that the original formula is satisfied by the class of standard models corresponding to KT4.

3. RECURSIVE RESOLUTION METHOD FOR MODAL LOGIC

3.1 Overview

This section is devoted to the recursive resolution method as developed by Chan (1987). The notion of recursion stems from the fact that modal formulas are recursive and therefore the method that is employed to resolve two modal clauses is a recursive method. The method was initially developed for the system KT4, and it is in this context that we will describe the method.

In order to conclude $\Gamma \square_C A$ using recursive resolution we need to show that the set $\Gamma \cup \{\neg A\}$ is unsatisfiable, that is, if we succeed in showing that $\Gamma \cup \{\neg A\}$ is unsatisfiable then we may conclude $\Gamma \square_C A$ where $\square$ is local logical entailment (see definition 24), Γ is any finite satisfiable set of formulas, C the corresponding class of standard models for the normal system of modal logic KT4, and A is any formula.

An outline of the method can be given as follows (terminology used in the outline is defined in the ensuing sections):

1. The finite set Γ ($\Gamma = \{A_1, A_2, \ldots, A_n\} \cup \{\neg A\}$) of well formed modal formulas that are being tested for unsatisfiability (see definition 21) are converted to a set Γ' of modal normal forms called m(modal)-clauses.

2. An algorithm is applied to the clauses to determine if they are modal-resolvable.

3. A modal resolvent is generated if two clauses are modal-resolvable.

4. Applying steps 2 and 3 repeatedly the empty clause will eventually be generated if the initial set was unsatisfiable. Otherwise the empty clause will not be derived.

3.2 Syntax and Semantics

As discussed in section 2.3.2, the main issue with modal resolution is ensuring that complementary literals are evaluated at the same world. For example, in the modal formula $P \wedge \Diamond\neg P$, P and $\neg P$ are complementary literals but they are not true at the same world and can therefore not be resolved with each other. Chan (1987) resolves this issue by replacing the $\Diamond$ with a new connective $\Diamond_x$ where x is any integer. The justification for this replacement follows from the definition of $\Diamond$. Recall that for a model M=<W,R,V> and a world $\alpha \in W$ $\Box^{M}_{\alpha} \Diamond A$ if there exists a $\beta \in W$ such that $\alpha R\beta$ and $\Box^{M}_{\beta} A$. Informally it can be said that replacing the $\Diamond$ connective with the $\Diamond_x$ connective fixes the formula A at the world β, that is, the modal formulas are evaluated at the worlds at which their literals are true.

Formally the connective $\Diamond_x$ can be defined as follows:

Definition 40

The ***skolem-world function*** *f_x: $W \rightarrow W$, is a function which takes as its argument a world $\alpha \in W$ and gives as its value a world $\beta \in W$ such that $\alpha R\beta$, where x is some integer (Chan 1987:160).*

Definition 41

The ***skolem world*** *is an object of the set of possible worlds W defined by applying one of the skolem-world functions $\{f_1, f_2, ..., f_x, ...\}$ to a member of W (Chan 1987:160).*

Definition 42

The subscripts $\{1,2, ..,x\}$ in the connectives $\Diamond_1$, $\Diamond_2$, ..., $\Diamond_x$, ..., are called ***world indexes*** *(Chan 1987:160).*

Definition 43

For any well formed formula A, model M=<W,R,V> and $\alpha \in W$: $\Box^{M}_{\alpha} \Diamond_x A$ if there exists $f_x(\alpha) \in W$ such *that $\alpha R f_x(\alpha)$ and $\Box^{M}_{f_x(\alpha)} A$ (Chan 1987:160).*

3.3 Rules

3.3.1 *Conversion to Normal Form*

As with ordinary resolution it is useful to reduce the formulas in the system to some form of minimum structure. This reduction is performed by means of the following conversion rules defined by Chan (1987).

1. Remove the → and ↔ as defined in chapter 2 from A.

2. Distribute any negations in A until they immediately precede their corresponding atoms using the following rules:

   2.1 $\neg\neg A$ is replaced by A

   2.2 $\neg(A \wedge B)$ is replaced by $\neg A \vee \neg B$

   2.3 $\neg(A \vee B)$ is replaced by $\neg A \wedge \neg B$

   2.4 $\neg\Box A$ is replaced by $\Diamond\neg A$.

   2.5 $\neg\Diamond A$ is replaced by $\Box\neg A$

3. Replace each $\Diamond$ in A by a new Skolem connective $\Diamond_x$, where x is the next positive integer not occurring in the set of world indexes.

4. Use the following rules to push the modal connectives as far as possible into the logical connectives.

   4.1 $\Diamond_x(A \vee B)$ is replaced by $(\Diamond_x A \vee \Diamond_x B)$

   4.2 $\Diamond_x(A \wedge B)$ is replaced by $(\Diamond_x A \wedge \Diamond_x B)$

   4.3 $\Box(A \wedge B)$ is replaced by $(\Box A \wedge \Box B)$

5. Use the following rule to distribute a disjunction into a conjunction:

   $(A \vee (B \wedge C))$ is replaced by $((A \vee B) \wedge (A \vee C))$

6. Flatten the conjunctive using the following equivalence:

   6.1 $(A \wedge (B \wedge C))$ is replaced by $(A \wedge B \wedge C)$

7. Split the conjuncts (if any) into clauses.

Definition 44

A formula that has been transformed by means of applying the above rules is a clause in ***modal normal form*** *referred to as a* ***m-clause*** *(Chan 1987:162).*

One note of caution: Chan (1987) stresses the importance of step 3 only taking place **after** step 1 and 2. This is because the main proposition defined below does not hold if any one of the connectives $\neg$, $\leftrightarrow$ and $\rightarrow$ precedes the connective $\Diamond_x$.

For example, the formula $\Box(P \rightarrow Q) \rightarrow (\Box P \rightarrow \Box Q)$ can be converted to modal normal form as follows.

1.	$\Box(P \rightarrow Q) \rightarrow (\Box P \rightarrow \Box Q)$	
2.	$\neg\Box(\neg P \vee Q) \vee (\neg\Box P \vee \Box Q)$	Step 1
3.	$\Diamond\neg(\neg P \vee Q) \vee (\Diamond\neg P \vee \Box Q)$	Step 2
4.	$\Diamond(\neg\neg P \wedge \neg Q) \vee (\Diamond\neg P \vee \Box Q)$	Step 2
5.	$\Diamond(P \wedge \neg Q) \vee (\Diamond\neg P \vee \Box Q)$	Step 2
6.	$\Diamond_1(P \wedge \neg Q) \vee (\Diamond_2\neg P \vee \Box Q)$	Step 3
7.	$(\Diamond_1 P \wedge \Diamond_1\neg Q) \vee (\Diamond_2\neg P \vee \Box Q)$	Step 4
8.	$(\Diamond_1 P \vee \Diamond_2\neg P \vee \Box Q) \wedge (\Diamond_1\neg Q \vee \Diamond_2\neg P \vee \Box Q)$	Step 5

The following m-clauses are derived from step 8: $(\Diamond_1 P \vee \Diamond_2\neg P \vee \Box Q)$, $(\Diamond_1\neg Q \vee \Diamond_2\neg P \vee \Box Q)$

Definition 45

A ***m-literal*** *is a literal preceded by its binding modalities with respect to a m-clause. (i.e. the variable is under their scope) (Chan 1987:164).*

For example in the m-clause $(\Diamond_1 P \vee \Diamond_2\neg P \vee \Box Q)$, the m-literals are $\Diamond_1 P$, $\Diamond_2\neg P$ and $\Box Q$.

Definition 46

A pair of m-literals such that their respective literals are negations to each other are referred to as a ***complementary pair of m-literals*** *(Chan 1987:164).*

For example in the pair of m-literals $\Box Q$ and $\Diamond_1\neg Q$ the literals Q and $\neg Q$ are negations of each other and therefore $\Box Q$ and $\Diamond_1\neg Q$ are a complementary pair of m-literals.

3.3.2 *Modal Resolution Criterion*

In the previous section we looked at how to derive corresponding m-clauses. In this section an algorithm is described that can determine whether any two m-literals can be resolved with each other.

Generally when two m-clauses have been identified that contain two complementary m-literals the m-literals are extracted from the clauses with *all* their preceding modal connectives. For example, in the clause $\Box(\neg P \vee \Diamond_1\neg Q)$ the m-literals are $\Box\neg P$ and $\Box\Diamond_1\neg Q$ and in the clause $\Diamond_3\Box(Q \vee \Box\neg R)$ the respective m-literals are $\Diamond_3\Box Q$ and $\Diamond_3\Box\Box\neg R$. Once the complementary literals have been extracted they are passed to the algorithm which determines if they are, in fact, complementary. If they are then the algorithm succeeds, else the algorithm fails. For example, in the clauses $\Box(\neg P \vee \Diamond_1\Diamond_2 R)$ and $\Diamond_3 P$ the complementary m-literals are: $\Box\neg P$ and $\Diamond_3 P$ and these m-literals would be passed to the algorithm. In this case the reader will note that the algorithm succeeds since the m-literals $\Box\neg P$ and $\Diamond_3 P$ are indeed complementary, that is, there does not exist a reflexive transitive model M=<W, R, V> such that both $\Box\neg P$ and $\Diamond_3 P$ are satisfied.

The algorithm is based on 18 pattern matching lemmas which compare different sequences of $\Box$ and $\Diamond$ for the two complementary literals. This algorithm is known as the Modal Resolution Criterion Algorithm (MRC).

Modal Resolution Criterion Algorithm (Chan 1987:164)

```
BASE TEST(A,B)
    IF A is a literal THEN
        IF B is a literal THEN
            Return SUCCEED;
        END IF
        IF some $\Diamond_a$ occurs in B THEN
            Return FAIL;
        END IF
    END IF
    IF B is a literal THEN
        IF some $\Diamond_a$ occurs in A THEN
            Return FAIL;
        ELSE
            Return SUCCEED;
        END IF
    END IF
    Return NIL;
END BASE TEST
```

```
TEST1(A,B)
        F1 = first modality of A;
        F2 = fist modality of B;
        IF F1 and F2 are identical $\Diamond_a$ THEN
                A1 = rest of A;
                B1 = rest of B;
                result = result of BASE TEST(A1, B1);
                IF result = FAIL THEN
                        return FAIL;
                ELSE IF result = SUCCEED THEN
                        return SUCCEED;
                ELSE
                        return TEST1(A1, B1);
                END IF
        END IF
        IF F1 = $\Diamond_a$ and F2 = $\Diamond_b$ and a differs from b THEN
                Return NIL
        END IF
        IF F1 = $\Box$ THEN
                Return the list(A,B);
        END IF
        IF F2 = $\Box$ THEN
                Return the list(B,A);
        END IF
END TEST1(A,B)

TEST2(A,B)
        L1 = last modality of A;
        R1 = rest of A;
        L2 = last modality of B;
        R2 = rest of B;
        IF L1 AND L2 are identical $\Diamond_a$ THEN
                result = result of BASE TEST(R1, R2);
                IF result = FAIL THEN
                        Return FAIL;
                ELSE IF result = SUCCEED THEN
                        Return SUCCEED;
                ELSE
                        Return TEST2(R1, R2);
                END IF
        END IF
        IF either L1 = $\Box$ or L2 = $\Box$ (or both) THEN
                Return the list (R1, R2);
        END IF
        IF L1 = $\Diamond_a$ AND L2 = $\Diamond_b$ and a differs from b THEN
                Return FAIL
        END IF
END TEST2(A,B)
```

```
TEST3(A,B)
    IF A does not contain any $\Diamond_a$ THEN
        return SUCCEED;
    END IF
    IF B is a literal THEN
        return FAIL
    END IF
    FA = the leftmost $\Diamond_c$ of A for any integer c;
    HB = first modality of B;
    RB = rest of B;
    IF FA is not equal to HB THEN
        Return TEST3(A, RB);
    END IF
    RA = tail sublist of A after the $\Diamond_c$;
    IF RA does not contain any $\Diamond_a$ THEN
        return SUCCEED;
    END IF
    IF RB is a literal THEN
        return FAIL;
    END IF
    result = TEST1(RA, RB);
        IF result = NIL THEN
        Return TEST3(A, RB)
    ELSE IF result = SUCCEED THEN
        Return SUCCEED;
    ELSE IF result = FAIL
        Return FAIL;
    END IF
    A' = first element of result;
    B' = second element of result;
    Return TEST3(A', B');
END TEST3(A,B)
```

```
MAIN(A,B)
    result = result of BASE TEST(A,B)
    IF result = FAIL THEN
        Return FAIL;
    ELSE IF result = SUCCEED THEN
        Return SUCCEED;
    ELSE
        result = result of TEST1(A,B);
    END IF

    IF result = FAIL OR NIL THEN
        Return FAIL;
    ELSE IF result = SUCCEED THEN
        Return SUCCEED;
    ELSE
        (A,B) = result;
        result = result of TEST2(A,B);
    END IF
    IF result = FAIL THEN
        Return FAIL;
    ELSE IF result = SUCCEED THEN
        Return SUCCEED;
    ELSE
        (A,B) = result;
    END IF
    IF □ occurs in A THEN
        IF □ occurs in B THEN
            Return SUCCEED;
        ELSE
            Return TEST3(A,B);
        END IF
    END IF
    Return TEST3(B, A);
END MAIN
```

Definition 47

A pair of complementary m-literals L_A and L_B, satisfying the MRC algorithm, are called a pair of ***MRC m-literals*** *(Chan 1987:170).*

Definition 48

Two m-clauses {A, B} are ***modal-resolvable*** *if and only if two m-literals {L_A, L_B} of A and B respectively are MRC M-literals (Chan 1987:170).*

3.3.3 *Resolution Rules*

Once a pair of MRC m-literals have been identified their corresponding m-clauses can be resolved in order to obtain a modal resolvent. This modal resolvent is determined by applying two types of rules known as deletion and special recursion rules. These rules are presented here without justification, since justification for these rules can be found in the section that deals with soundness and completeness. The deletion rules are described first:

Deletion Rules:

Deletion rules are essentially operational rules that are used to determine which modal connectives can be deleted from two m-literals (Chan 1987:170).

Definition 49

*A set of two m-literals obtained from applying a deletion rule to the m-literals L_A and L_B are denoted by **De(L_A, L_B)**.*

Deletion Rule 1: If L_A is $\Diamond_x A'$ and L_B is $\Diamond_x B'$ for any well formed formula A' and B', then delete the skolem operator $\Diamond_x$ from both L_A and L_B, i.e. De(L_A, L_B) = {A', B'}.

Deletion Rule 2: For all other cases, delete the first operator from either L_A or L_B, but not both, in such a way that the two resulting m-clauses in De(L_A, L_B) obtained are MRC m-literals.

Definition 50

In a m-clause B,

1. *A **core-disjunct** denoted C(B), relative to an occurrence of a literal L is the disjunct B such that L occurs in B.*

2. *A **side-disjunct**, denoted S(B), relative to an occurrence of a literal L is the disjunction consisting of all disjuncts in B such that the occurrence of the literal L does not occur in any these disjuncts. (Chan 1987:172.)*

For example consider the clause B: ($\Diamond_1 P \vee \Box\neg R \vee \Diamond_2\Box\neg Q$). Here we have the following clauses for S(B) and C(B) relative to the literal P. S(B): ($\Box\neg R \vee \Diamond_2\Box\neg Q$) and C(B): ($\Diamond_1 P$), that is, B is a m-clause, which is the disjunction of C(B) $\vee$ S(B).

The deletion rules can be applied directly to the core-disjuncts of the m-clauses A and B, disregarding the side-disjuncts, in order to obtain the result Del(A, B).

Definition 51

Del(A,B): *Two m-clauses formed after some modalities are deleted from m-clauses {A, B} according to the following. If L_A and L_B are two MRC m-literals in m-clauses A and B respectively then delete modalities from {A, B} according to the deletion rules as if {A, B} are just {L_A, L_B}.*

For example, if we have the following two m-clauses: A: $\Diamond_1\Box(Q \vee \Diamond_2 P)$ and B: $\Box(\Diamond_3\neg P \vee R)$ then we could form Del(A,B) by defining $L_A = \Diamond_1\Box\Diamond_2 P$ and $L_B = \Box\Diamond_3\neg P$, and only apply the deletion rules to {L_A, L_B}. Note that the deletion rules are applied to A and B as if no side-disjunct nor $\vee$ occurs in A or B.

Special Recursion Rules

Special recursion rules use the deletion rules in a specified way to obtain a *modal resolvent* for any two m-clauses.

Definition 52

Given two m-clauses {A,B}, if there exists a pair of complementary MRC m-literals {L_A, L_B} respectively in A and B, then some Special Recursive Resolution (SRR) rules can be applied to {A,B} to construct some Modal Resolvent (MR) of {A,B} denoted by MR(A,B). We call ***MR(A,B)*** *a* ***modal resolvent*** *of A and B. (Chan 1987:172.)*

Chan (1987:172, 173) defines the following five SRR rules, where {A, B} are two m-clauses:

SRR1: If {A,B} are just two complementary MRC m-literals, then the MR is empty (denoted by $\bot$).

SRR2: If one of the m-clauses, say A, is a disjunction of a side-disjunct S(A) and a core-disjunct C(A), then MR(A,B) = (S(A) $\vee$ MR(C(A),B)).

SRR3: If one of the m-clauses, say A, is a m-literal and B is $\Box$B', then MR(A,B) = MR(A,B'), where B' = B without the connective $\Box$.

SRR4: If one of the m-clauses, say A, is $\Diamond_x$A' and B is $\Box$B', then we apply the deletion rules to delete a modality from either A or B.

4.1 If we delete the first $\Box$ from B, then MR(A,B) = MR(A,B'), where B' is B without the modality $\Box$.

4.2 If we delete the first $\Diamond_x$ from A we apply the rule SRR5.

SRR5: For all the other cases, we delete some modalities from {A,B} by the deletion rules as if applying the deletion rules to the complementary MRC m-literals {L_A, L_B} in {A,B}. Let * be the deleted modality from A or B or both, let {Delet(A), Delet(B)} be the two m-clauses in Del(A,B) obtained after the deletion. Then MR (A,B) = *(MR(Delet(A), Delet(B))), that is, we append the * to the recursive part.

Note that the MR(A,B) derived from the above rules may not be unique. This is because there are a number of ways to apply the above rules. For example, if both the m-clauses A and B are of the type $\Box$A' and $\Box$B', SSR3 could be applied to either A or B.

3.4 Soundness and Completeness

No formal soundness and completeness results are provided by Chan (1987) since formal proofs may be found in previous research articles published by Chan in 1985. He does however provide the following justifications which are stated without proof.

3.4.1 *Justification for the Conversion Rules*

Main Proposition

Let Γ be a set of formulas and A any formula $\in \Gamma$. Let A' be the result of applying steps 1, 2 and 3 of the conversion rules to A. Γ is unsatisfiable if and only if (Γ - A) $\cup$ {A'} is unsatisfiable (Chan 1987:163).

The conversion rules guarantee by the main proposition that if we start out with an unsatisfiable set say Γ, and the conversion rules are applied to Γ to obtain Γ', then Γ' will still be an unsatisfiable set once the conversion has been completed.

3.4.2 *Justification for the MRC Algorithm*

Theorem 42

A pair of complementary m-literals A and B are unsatisfiable iff they satisfy the MRC (Chan 1987:166).

Proof: Essentially the proof includes 18 lemmas which use a pattern matching method to examine 18 different forms (patterns) of m-literals. Each of these patterns are proved to be satisfiable, unsatisfiable or its satisfiability has to be determined recursively. We refer the reader to Chan (1985) for a formal proof. □

3.4.3 *Justification for the Deletion Rules*

Justification follows from the following two lemmas.

Lemma 1
For any pair of m-literals $\{L_A, L_B\}$ satisfying the MRC algorithm, it is always possible to delete a leftmost modality from either L_A or L_B or both to get a set $De(L_A, L_B)$ such that $De(L_A, L_B)$ is a pair of m-literals that satisfies the MRC algorithm (Chan 1987:171).

Lemma 2
Applying the deletion rules repeatedly to any two MRC m-literals A and B generates a sequence of pairs of m-literals $De^1(A,B)$, $De^2(A,B)$, …, $De^n(A,B)$ such that each satisfies MRC, and both the m-clauses in $De^n(A,B)$ are complementary literals for some finite n (Chan 1987:171).

3.4.4 *Soundness*

Theorem 43

A MR(A, B) is locally logically entailed (see definition 24) by the m-clauses {A, B} (Chan 1987:173).

Roughly what this theorem is saying is that $\{A, B\}\Box_C$ MR(A, B) where C is the class of reflexive transitive standard models and $\Box$ is local logical consequence, that is, for every M=<W, R, V> in C and for every $\alpha \in W$, if $\Box_{\alpha}^{M} A \wedge B$ then $\Box_{\alpha}^{M}$MR(A, B), that is, if M satisfies the two m-clauses then M will also satisfy their modal resolvent. In other words the SSR rules are sound.

3.4.5 *Completeness*

Chan (1987:178) notes that completeness is complicated by the fact that the SRR method only generates some of the m-clauses. For example, if A is the m-clause $\Diamond_1\Box(P \vee \Box(\Diamond_2 P \vee Q))$ and B is the m-clause $\Box(\neg P \vee \Diamond_3\Box(\neg P \vee R))$ then although the m-literals $\{\Diamond_1\Box P, \Diamond_1\Box\Box\Diamond_2 P\}$ in A satisfy the MRC with each of the m-literals $\{\Box\neg P, \Box\Diamond_3\Box\neg P\}$ in B, the SSR method would not compare the m-literals in A with each of the m-literals in B. It may just choose the first m-literal in B or the second but not both.

There is, however, a more advanced method known as the General Resolution Recursive method of which SSR is a subset. This method is known to generate modal resolvents for several literals simultaneously and the reader is referred to Chan (1985) for further details.

3.5 Applications

This resolution system is demonstrated by means of the following example.

Consider the following two formulas: $\Box(P \rightarrow \Diamond(\neg Q \wedge \Diamond R))$ and $\Diamond(\Box(\neg Q \rightarrow \Box\neg R) \wedge P)$. We attempt to show that these two formulas are unsatisfiable. Before we can do so the formulas have to be converted to their corresponding m-clauses:

1. $\Box(P \rightarrow \Diamond(\neg Q \wedge \Diamond R))$
2. $\Box(\neg P \vee \Diamond(\neg Q \wedge \Diamond R))$ Step 1
3. $\Box(\neg P \vee \Diamond_1(\neg Q \wedge \Diamond_2 R))$ Step 3
4. $\Box(\neg P \vee (\Diamond_1\neg Q \wedge \Diamond_1\Diamond_2 R))$ Step 4
5. $\Box((\neg P \vee \Diamond_1\neg Q) \wedge (\neg P \vee \Diamond_1\Diamond_2 R))$ Step 5
6. $\Box(\neg P \vee \Diamond_1\neg Q) \wedge \Box(\neg P \vee \Diamond_1\Diamond_2 R)$ Step 4

The m-clauses corresponding to $\Box(P \rightarrow \Diamond(\neg Q \wedge \Diamond R))$ are therefore $\Box(\neg P \vee \Diamond_1\neg Q)$, $\Box(\neg P \vee \Diamond_1\Diamond_2 R)$.

Converting $\Diamond(\Box(\neg Q \rightarrow \Box\neg R) \wedge P)$ to clause form:

1.	$\Diamond(\Box(\neg Q \rightarrow \Box\neg R) \wedge P)$	
2.	$\Diamond(\Box(\neg\neg Q \vee \Box\neg R) \wedge P)$	Step 1
3.	$\Diamond(\Box(Q \vee \Box\neg R) \wedge P)$	Step 2
4.	$\Diamond_3(\Box(Q \vee \Box\neg R) \wedge P)$	Step 3
5.	$\Diamond_3\Box(Q \vee \Box\neg R) \wedge \Diamond_3 P$	Step 4

The m-clauses are therefore $\Diamond_3\Box(Q \vee \Box\neg R)$, $\Diamond_3 P$.

Thus the proof starts with the following m-clauses:

1. $\Box(\neg P \vee \Diamond_1\neg Q)$
2. $\Box(\neg P \vee \Diamond_1\Diamond_2 R)$
3. $\Diamond_3\Box(Q \vee \Box\neg R)$
4. $\Diamond_3 P$.

From these m-clauses the reader should note that the set of world-indexes is the set {1, 2, 3}.

In the proof below the centre column contains the recursive modal resolvent denoted by the function MR(C_1, C_2) where C_1 and C_2 are the two m-clauses that are being resolved. The right most column contains the SSR rules that were applied to the two m-clauses C_1 and C_2. The modal resolvent derived in terms of the relevant SSR rules is appended to the left hand side expression, if any. (Chan 1987).

Consider the m-clauses $\Box(\neg P \vee \Diamond_1\neg Q)$, $\Diamond_3 P$. The corresponding complementary m-literals are $\Box\neg P$ and $\Diamond_3 P$. Calling the MRC($\Box\neg P$, $\Diamond_3 P$) we find that these literals are MRC m-literals, i.e. the algorithm succeeds. It follows therefore that their corresponding clauses can be resolved with each other as follows:

5.	MR($\Box(\neg P \vee \Diamond_1\neg Q)$, $\Diamond_3 P$)	
6. $\Diamond_3$	MR($\Box(\neg P \vee \Diamond_1\neg Q)$, P)	SSR4 (4.2)
7. $\Diamond_3$	MR(($\neg P \vee \Diamond_1\neg Q$), P)	SSR3
8. $\Diamond_3(\Diamond_1\neg Q \vee$	MR($\neg$P, P))	SRR2
9. $\Diamond_3(\Diamond_1\neg Q \vee$	$\perp$)	SRR1
10.$\Diamond_3\Diamond_1\neg Q$		

Step 6 is the result of applying the SSR4 (4.2) rule to the m-clauses $\Box(\neg P \vee \Diamond_1\neg Q)$ and $\Diamond_3 P$. Step 7 is the result of applying the SSR3 rule to the m-clauses $\Box(\neg P \vee \Diamond_1\neg Q)$ and P. Applying the SSR2 rule to the m-clauses $(\neg P \vee \Diamond_1\neg Q)$ and P, where the side disjunct with respect to the literal P is $\Diamond_1\neg Q$ yields the resolvent $\Diamond_1\neg Q \vee MR(\neg P, P)$. Appending this result to the left hand side gives the result in step 8. By SRR1 $MR(\neg P, P)$ is $\bot$ and therefore the final clause is $\Diamond_3(\Diamond_1\neg Q \vee \bot)$ which is the same as $\Diamond_3\Diamond_1\neg Q$ since $\bot$ denotes the empty clause.

The clauses $\Box(\neg P \vee \Diamond_1\Diamond_2 R)$ and $\Diamond_3 P$ are now resolved, since the m-literals $\Box\neg P$ and $\Diamond_3 P$ are complementary MRC m-literals.

11.	$MR(\Box(\neg P \vee \Diamond_1\Diamond_2 R), \Diamond_3 P)$	
12. $\Diamond_3$	$MR(\Box(\neg P \vee \Diamond_1\Diamond_2 R), P)$	SSR4 (4.2)
13. $\Diamond_3$	$MR((\neg P \vee \Diamond_1\Diamond_2 R), P)$	SSR3
14. $\Diamond_3(\Diamond_1\Diamond_2 R \vee$	$MR(\neg P, P))$	SRR2
15. $\Diamond_3(\Diamond_1\Diamond_2 R$	$\bot)$	SRR1
16. $\Diamond_3\Diamond_1\Diamond_2 R$		

Step 10 is the result of applying the SSR4 (4.2) rule to the clauses $(\Box(\neg P \vee \Diamond_1\Diamond_2 R), \Diamond_3 P)$. Step 11 is the result of applying the SSR3 rule to the clauses $(\Box(\neg P \vee \Diamond_1\Diamond_2 R), P)$. Step 12 is the result of applying the SSR2 rule to the clauses $((\neg P \vee \Diamond_1\Diamond_2 R), P)$ where the side disjunct with respect to the literal P is $\Diamond_1\Diamond_2 R$. Since by SSR1 $MR(\neg P, P)$ is $\bot$ the modal-resolvent is just $\Diamond_3\Diamond_1\Diamond_2 R$.

Resolving the clauses $\Diamond_3\Box(Q \vee \Box\neg R)$ and $\Diamond_3\Diamond_1\Diamond_2 R$ (from step 16). In this instance the complementary MRC m-literals are $\Box\neg R$ and $\Diamond_3\Diamond_1\Diamond_2 R$.

17.	$MR(\Diamond_3\Box(Q \vee \Box\neg R), \Diamond_3\Diamond_1\Diamond_2 R)$	
18. $\Diamond_3$	$MR(\Box(Q \vee \Box\neg R), \Diamond_1\Diamond_2 R)$	SRR4 (4.2)
19. $\Diamond_3\Diamond_1$	$MR(\Box(Q \vee \Box\neg R), \Diamond_2 R)$	SRR4 (4.2)
20. $\Diamond_3\Diamond_1$	$MR((Q \vee \Box\neg R), \Diamond_2 R)$	SRR4 (4.1)
21. $\Diamond_3\Diamond_1(Q \vee$	$MR(\Box\neg R, \Diamond_2 R))$	SRR2 (S(R) = Q)
22. $\Diamond_3\Diamond_1(Q \vee$	$\bot)$	SRR1
23. $\Diamond_3\Diamond_1 Q$		

Similar arguments hold here as discussed above.

Resolving $\Diamond_3\Diamond_1 Q$ (from step 20) and $\Diamond_3\Diamond_1\neg Q$ (from step 9), since $\Diamond_3\Diamond_1 Q$ and $\Diamond_3\Diamond_1\neg Q$ are complementary MRC m-literals.

24.	$MR(\Diamond_3\Diamond_1 Q, \Diamond_3\Diamond_1\neg Q)$	
25	$\bot$	SRR1

Since $\bot$ was derived it can be concluded that the original formulas where unsatisfiable.

A final note on this example. There were many more modal-resolvents that could have been generated from this example. The interested reader may want to refer to a similar example in Chan (1987) for the other variations.

4. *LINEAR MODAL DEDUCTIONS*

4.1 Overview

In this section the resolution system devised by L.F. del Cerro and A Herzig (1988) is introduced. This resolution system is similar to Chan's (1985) resolution system introduced in the previous section, since both resolution systems operate on clauses and both resolution systems skolemise the modal connective $\Diamond$. There are, however, some differences. One is that Chan's (1985) method is of a recursive nature and del Cerro & Herzig's (1988) resolution system is of a linear nature. Also the way del Cerro & Herzig (1988) skolemise the $\Diamond$ is different to that of Chan (1985). Del Cerro & Herzig (1988) only presented formal results for the normal system of modal logic KT4 although they argue that similar results would hold for the other systems of modal logic. Their resolution system will, therefore, only be discussed in terms of the system KT4 and the reader is referred to del Cerro & Herzig's (1988) work for notes with regards to the other systems.

Using this resolution system we can show $\Gamma\Box_C A$ as follows. Let Γ be any satisfiable set of formulas and A any formula, and C the class of transitive and reflexive standard models (since we are only dealing with KT4). If we succeed in finding a proof for $\Gamma \cup \{\neg A\}$ using del Cerro & Herzig's (1988) resolution system then we can conclude $\Gamma\Box_C A$ where $\Box$ is defined as in definition 24, i.e. local logical entailment.

4.2 Syntax

Del Cerro and Herzig (1988) designed a Skolem technique whereby they replaced the ◊ by a Skolem constant i. Let L denote the modal language as presented in chapter 3 then the language LI is the modal language with a set of I of Skolem constants where the unary modal connectives are □ and <i>, for every $i \in I$. Formulas in this language are described as those in definition 18 except for: if A is a formula then <i>A, for every $i \in I$ are formulas.

A formula A of L can be mapped into a formula A^I of LI by replacing each occurrence of ◊ in A by <i> where i is a new constant (so in A^I every <i> occurs only once, and I is the set of new constants).

For example, if A is $\Box(\Diamond P \vee \Diamond(Q \wedge \neg T))$ then A^I will be $\Box(\langle 1\rangle P \vee \langle 2\rangle(Q \wedge \neg T))$ and $I = \{1,2\}$.

4.3 Semantics

The semantics for LI is defined in terms of a model MI = <W,R,V,FI> where $FI = \{f_i \mid i \in I\}$ is a set of functions from W into W, where $f_i \subseteq R$ for every $i \in I$. Definitions of W, R, V remain the same. We define the truth of a formula A at a world $\alpha \in W$ as $\square_{\alpha}^{MI}$ <i>A if and only if $\square_{f_i(\alpha)}^{MI}$ A. Satisfaction and validity are defined for LI as follows:

Definition 53

We say that MI=<W,R,V,FI> ***satisfies*** *A or MI is a model for A if and only if there is an $\alpha \in W$ such that $\square_{\alpha}^{MI}$ A, and A is* ***valid*** *if and only if for every MI and every $\alpha \in W$, we have $\square_{\alpha}^{MI}$ A (del Cerro & Herzig 1988:488).*

The following theorem guarantees that if a formula A of L is mapped into a formula A^I of LI (see section 4.2) then A is satisfiable in a model for L if and only if A^I is satisfiable in a model for LI. In other words the mapping from A to A^I is sound.

Theorem 44

Let A be a modal formula of L. A is satisfiable in a model for L iff A^I is satisfiable in a model for LI.

(See del Cerro & Herzig 1988:490 for a proof.)

4.4 Rules

The rules applicable to this system can be summarised as follows and are treated individually below.

1. Conversion rules.
2. Inference rules.
3. Simplification rules.
4. Resolution rules.

4.4.1 *Conversion Rules*

In order to apply the rules to the formulas, using this resolution system, the formulas have to be converted to clausal normal form, defined as follows:

Definition 54

A formula of LI is said to be in ***clausal form*** *if it is a disjunction in which each disjunct is either:*

- *a literal,*
- *$\Box C$, where C is a formula in clausal form or*
- *$<i>$ C where $i \in I$ and C is a formula in clausal form.*

A formula is said to be in ***clausal normal form*** *if it is a conjunction in which each conjunct is in clausal form (del Cerro & Herzig 1988:490).*

For example $<1>(P \vee \Box(Q \vee <2>\neg Q)) \wedge Q$, is in clausal normal form.

We will denote a conjunction $A_1 \wedge A_2 \wedge \ldots \wedge A_n$ by means of the set $\{A_1, A_2, \ldots, A_n\}$. We will also refer to each conjunct as a *clause*.

It follows that before we can use this resolution system on a formula A of L we have to map A to A^I of LI and then convert A^I to an equivalent formula A' in clausal normal form. In section 4.2 we showed how the mapping from A to A^I was possible. In the theorem below the mapping from A^I to A' is confirmed.

Theorem 45

There is an effective procedure for the construction of any given formula A of LI to an equivalent formula A' in clausal normal form.

Proof:

The proof is by induction on the modal degree of A.

The ***modal degree*** *$d(A)$ of a formula A is defined as follows:*

- *If A is a literal, $d(A) = 0$*
- *$d(A * B) = \max(d(A), d(B))$ if $*$ is $\vee$, $\wedge$, $\rightarrow$ or $\leftrightarrow$*
- *$d(\neg A) = d(A)$*
- *$d(* A) = d(A) + 1$ if $*$ is $\Box$ or $\langle i \rangle$, where $i \in I$.*

If $d(A) = 0$ the proof is obtained using the classical procedure. If $A = \Box B$ then we normalise B and replace instances of $\Box(A \wedge B)$ by $\Box A \wedge \Box B$. If $A = \langle i \rangle B$ then we normalise B and replace instances of $\langle i \rangle(A \wedge B)$ with $\langle i \rangle A \wedge \langle i \rangle B$ and instances of $\langle i \rangle(A \vee B)$ with $\langle i \rangle A \vee \langle i \rangle B$. Otherwise A is a classical combination of modal formulas, then we can normalise the latter and the apply the classical procedure. (del Cerro & Herzig 1988:491.)□

Although not mentioned in the theorem above this resolution system assumes that the connective $\neg$ appears immediately before the atom. For instance instead of $\neg\Diamond P$ we work with $\Box\neg P$.

4.4.2 *Inference Rules*

Inference rules are rules of the form C1, C2 $\Rightarrow$ C (read C is inferred from C_1 and C_2) and are defined by means of the following formal system (del Cerro & Herzig 1988:491):

axiom:	$A, \neg A \Rightarrow \bot$
$\vee$-rule:	if $A, B \Rightarrow C$ then $A \vee D, B \vee E \Rightarrow C \vee D \vee E$
T-rule:	if $A, B \Rightarrow C$ then $\Box A, B \Rightarrow C$
$\Box\Box$-rule:	if $\Box A, B \Rightarrow C$ then $\Box A, \Box B \Rightarrow \Box C$
$\Box$<i>-rule:	if $\Box A, B \Rightarrow C$ then $\Box A, <i>B \Rightarrow <i>C$
<i><i>-rule:	if $A, B \Rightarrow C$ then $<i>A, <i>B \Rightarrow <i>C$

For example given the two clauses $\Box(<1>P \vee Q)$ and $<1><2><1>\neg P$ we can derive a resolvent as follows. The formulas will be listed on the left hand side and justifications for their transformations will be listed on the right hand side.

1.	$P, \neg P \Rightarrow \bot$	Axiom
2.	$<1>P, <1>\neg P \Rightarrow <1>\bot$	1, <i><i>-rule
3.	$<1>P \vee Q, <1>\neg P \vee \bot \Rightarrow <1>\bot \vee Q \vee \bot$	2, $\vee$-rule
5.	$\Box(<1>P \vee Q), <1>\neg P \vee \bot \Rightarrow <1>\bot \vee Q \vee \bot$	4, T-rule
6.	$\Box(<1>P \vee Q), <2>(<1>\neg P \vee \bot) \Rightarrow <2>(<1>\bot \vee Q \vee \bot)$	5, $\Box$<i>-rule
7.	$\Box(<1>P \vee Q), <1><2>(<1>\neg P \vee \bot) \Rightarrow <1><2>(<1>\bot \vee Q \vee \bot)$	6, $\Box$<i>-rule

4.4.3 *Simplification Rules*

Simplification rules are rules of the form $C' \approx C$ (read C' can be simplified into C) and are defined as follows (del Cerro & Herzig 1988:492):

- $A \vee \bot \approx A$
- $*(\bot) \approx \bot$ if $*$ is $\Box$ or <i>, for $i \in I$.
- $A \vee A \vee B \approx A \vee B$

For example applying the simplification rules to the resolvent $\langle 1\rangle\langle 2\rangle(\langle 1\rangle\bot \vee Q \vee \bot)$ derived above yields:

8. $\langle 1\rangle\langle 2\rangle(\langle 1\rangle\bot \vee Q \vee \bot)$ from 7

9. $\langle 1\rangle\langle 2\rangle(\bot \vee Q \vee \bot)$ $*(\bot) \approx \bot$where $* = \Diamond$

10. $\langle 1\rangle\langle 2\rangle(Q \vee \bot)$ $A \vee A \vee B \approx A \vee B$

11. $\langle 1\rangle\langle 2\rangle Q$ $A \vee \bot \approx A$

We have shown that $\langle 1\rangle\langle 2\rangle Q$ is the resolvent of $\Box(\langle 1\rangle P \vee Q)$ and $\langle 1\rangle\langle 2\rangle(\langle 1\rangle\neg P \vee \bot)$. Now $\langle 1\rangle\langle 2\rangle(\langle 1\rangle\neg P \vee \bot)$ can be simplified to $\langle 1\rangle\langle 2\rangle\langle 1\rangle\neg P$ by means of the simplification rule $A \vee \bot \approx A$ and therefore it follows that $\langle 1\rangle\langle 2\rangle Q$ is the resolvent of $\Box(\langle 1\rangle P \vee Q)$ and $\langle 1\rangle\langle 2\rangle\langle 1\rangle\neg P$.

4.4.4 *The Resolution Rule*

Using the inference and simplification rules a resolvent for two clauses C_1 and C_2 can be derived as follows:

Definition 55 (Modal Resolution Rule)

Given two clauses C_1 and C_2, C is a resolvent of C_1 and C_2 if there is a C' such that $C_1, C_2 \Rightarrow C'$ and $C' \approx C$. The modal resolution rule can be denoted by: $\dfrac{C_1\ C_2}{C}$ *(del Cerro & Herzig 1988:491).*

Unlike other resolution rules discussed in this chapter this modal resolution rule is applied in a linear fashion to the clauses in the resolution proof, that is, the resolvent of the previous two clauses is always used in the next application of the resolution rule. This process is formally described as follows:

Definition 56

Given a set of clauses Γ, a ***linear deduction*** *of C_n from Γ with top clause C_0 is a deduction of the form:*

$$\frac{C_0 \quad D_0}{C_1}$$

$$\ldots$$

$$\frac{C_k \quad D_k}{C_{k+1}}$$

$$\ldots$$

$$\frac{}{C_n}$$

where $C_0 \in \Gamma$ and C_{k+1} is a resolvent of C_k and D_k and each D_k is either in Γ or is a Cj for some $j<k$. (del Cerro & Herzig 1988:492.)

Definition 57

A linear deduction of $\bot$ from Γ, a set of clauses, is called a ***refutation of Γ.***

4.5 Soundness and Completeness

Theorem 46 (Soundness)

If there is a linear refutation of Γ with top clause C_0 then Γ is unsatisfiable.

Proof: Essentially the proof is based on the fact that the inference rules are sound. Therefore if Γ is a satisfiable (see definition 53) set of formulas then when the inference rules are applied to Γ, every resolvent derived from Γ will also be satisfiable. It follows that ⊥ cannot be produced from this set. (Refer to del Cerro & Herzig 1988:493 for details of the proof). □

Theorem 47 (Completeness)

Let Γ be a set of clauses. If S is unsatisfiable and Γ-$\{C_0\}$ is satisfiable, then there is a linear refutation of Γ with top clauses C_0.

Proof: A formal proof can be found in del Cerro & Herzig (1988), we will provide a sketch only. Essentially the proof is based on the notion of trees very similar to that presented in chapter 2 for tableaux systems. In particular they make use of the following results which we state without proof:

1. **Lemma**

   Let S be a set of clauses of LI. If S is unsatisfiable then S has a closed tree.

2. **Lemma**

   Let S be a set of clauses. If S is unsatisfiable and S - $\{C_0\}$ is satisfiable, then S has a closed tree, such that in its root S is closed and S-$\{C_0\}$ is open.

3. **Upward Lemma**

   Given a node n of a tree and a subset n' of n, if n' is closed in n and n'-$\{C_0\}$ is open in n, then there is linear refutation of n' with top clause C_0.

To prove completeness: Let S be an unsatisfiable set of clauses. By 1 every tree for S is closed. Let n be a root of a tree for S. S will be closed in n and S-$\{C_0\}$ will be open in n by 2. Hence by the upward lemma there will be a linear refutation of S with top clause C_0.□

4.6 Applications

We demonstrate this resolution system by means of a simple example. Assume that the set Γ = $\{\neg(\Box P\rightarrow \Box\Box P)\}$ and that we want to show that Γ is unsatisfiable. From soundness we know that if there is a linear refutation of Γ then Γ is unsatisfiable. Hence we attempt to perform a linear refutation of this set as follows.

Distributing the negation and using the appropriate conversion rules described in section 4.4.1, Γ is converted to the following clause: $\{\Box P, <1><2>\neg P\}$ where the set I of Skolem constants is $\{1,2\}$.

A linear deduction can now be performed on this set. Note that the formulas are listed on the left hand side and the justifications for their transformation are listed on the right hand side.

1. $\Box P$
2. $\langle 1\rangle\langle 2\rangle\neg P$

Since there are only two clauses in this set we attempt to resolve them with each other. Looking at the two atoms we notice the complementary literals (P, $\neg P$). Hence we attempt to build the resolvent from these complementary literals using the corresponding inference and simplification rules as follows:

$P, \neg P \Rightarrow \bot$	Axiom
$\Box P, \neg P \Rightarrow \bot$	T-rule
$\Box P, \langle 2\rangle\neg P \Rightarrow \langle 2\rangle\bot$	$\Box\langle i\rangle$-rule
$\Box P, \langle 1\rangle\langle 2\rangle\neg P \Rightarrow \langle 1\rangle\langle 2\rangle\bot$	$\Box\langle i\rangle$-rule

The resolvent is therefore $\langle 1\rangle\langle 2\rangle\bot$, however we can simplify this resolvent as follows:

$\langle 1\rangle\langle 2\rangle\bot$	
$\langle 1\rangle\bot$	$*(\bot) \approx \bot$where $* = \langle 2\rangle$
$\bot$	$*(\bot) \approx \bot$where $* = \langle 1\rangle$

The resolvent of $\Box P$ and $\langle 1\rangle\langle 2\rangle\neg P$ is therefore $\bot$, that is, we have the following linear refutation:

1.	$\Box P$	
2.	$\langle 1\rangle\langle 2\rangle\neg P$	
3.	$\bot$	1,2

By soundness it follows that the original set Γ was unsatisfiable.

5. DESTRUCTIVE MODAL RESOLUTION

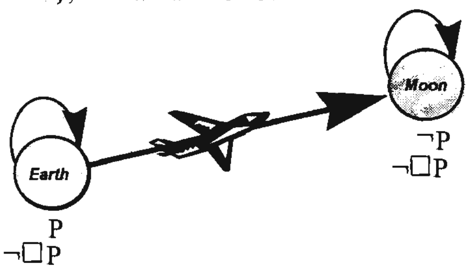
5.1 Overview

In some of the previous sections when we used resolution we converted the formulas into clausal form and then proceeded with the resolution reduction rules. In destructive modal resolution Fitting does the conversion as he progresses through the proof making use of the following two types of rules:

1. Replacement rules: - rules where a formula is removed and one or more new ones are added in its place.
2. Addition rules: - rules where formulas are added but none are removed.

Fitting's (1990a) resolution system is based on the notion of a context shift best described by means of the following example:

Assume that P denotes the sentence *Garfield eats Tweety* and that the formula □P → P is interpreted in a KT4 model M = <W, R, V> where W = {Earth, Moon}, R = {<Earth, Earth>, <Moon, Moon>, <Earth, Moon>}, V(P) = {Earth}, as illustrated below.



Clearly from this model we can see that P is true at the world denoted by *Earth* but not at the world denoted by *Moon.* (Tweety is therefore well advised to take the next space shuttle to the moon!), that is, as we shift from the Earth to the Moon we lose information. In destructive resolution the application of the rules result in a similar context shift which in turn results in the loss of information. It is because of this phenomenon that Fitting's (1990b) system is referred to as *destructive* resolution and is therefore only applicable to those logics whose standard models do not involve symmetry.

Although Fitting (1990a) covers additional normal systems we will only be looking at the systems KT4 and KT5 and we refer the interested reader to his work for the other systems. Also note that although Fitting doesn't cover the system KT4L we have taken the liberty to extend his method to the system KT4L.

Let Γ be the empty set and C any class of standard models corresponding to the normal system of modal logic Σ. Then we can conclude $\Gamma\square_C A$ (or just $\square_C A$) for any formula A if we find a destructive resolution proof for $\neg A$.

5.2 Syntax

As mentioned in chapter 2 (see section 3.3), Fitting (1990a) makes use of different types of formulas, in particular he distinguishes between four types: α, β, ν and π. The α and β type formulas were defined in chapter 2 and will therefore not be repeated here.

Necessary or ν formulas: - If a formula of type ν is true at a possible world, say γ, then ν_0 is true at every world accessible from γ.

ν	ν_0
$\Box A$	A
$\neg\Diamond A$	$\neg A$

Possible or π formulas:- If a formula of type π is true at a world say γ, then π_0 is true at some accessible world from γ.

π	π_0
$\neg\Box A$	$\neg A$
$\Diamond A$	A

We introduce a special notation for generalised disjunctions and conjunctions. We will treat $[A_1, A_2, \ldots, A_n]$ as synonymous with $A_1 \vee A_2 \vee \ldots \vee A_n$ and $\langle A_1, A_2, \ldots A_n \rangle$ as synonymous with $A_1 \wedge A_2 \wedge \ldots \wedge A_n$.

Definition 58

*If $A_1, A_2, \ldots, A_n$ are formulas, we will refer to $[A_1, A_2, \ldots, A_n]$ as a **clause** (Fitting 1990a:86).*

Note that this definition is not in accordance with definition 10 which defined the notion of a clause for the propositional case. The main difference is that in the propositional case a clause is a literal and in this case a clause is a formula.

Definition 59

If each C_i is a clause, we will refer to $\langle C_1, C_2, \ldots, C_n\rangle$ as a ***clause list*** *(Fitting 1990a:86).*

Definition 60

If $L_1, L_2, \ldots, L_n$ are clause lists, we call $[L_1, L_2, \ldots, L_n]$ a ***block*** *(Fitting 1990a:86).*

Definition 61

*We call a list containing the empty clause (denoted by []) **closed**, and a block closed if every clause list in it is closed. We call a block **closable** if a closed block can be derived by using the replacement and addition rules as defined below.*

Definition 62

*We say A is a **theorem** if the block* [⟨[¬A]⟩] *is closable (Fitting 1990a:86).*

5.3 Rules

As mentioned at the start of section 5 Fitting makes use of replacement and addition rules. We define the replacement rules first.

5.3.1 *Replacement Rules.*

Double Negation:

$$\frac{\neg\neg A}{A} \text{ where A is any formula.}$$

This rule says that any formula A that has the structure ¬¬A can be replaced by the formula A. For example the formula ¬¬□P can be replaced by the formula □P.

Conjunction:

$$\frac{[A_1,\ldots,A_n,\alpha]}{[A_1,\ldots,A_n,\alpha_1],[A_1,\ldots,A_n,\alpha_2]}$$

Where A is any formula, and α is any α-type formula (see chapter 2).

From this rule it should be clear that if a clause contains a α-type formula then the clause can be replaced by two additional clauses say C_1 and C_2 where C_1 contains the α_1 formula of α and C_2 contains the α_2 formula of α. For example, if we have $[\Box(P \wedge Q), \neg(P \vee Q)]$ then this clause can be replaced by the two clauses $[\Box(P \wedge Q), \neg P]$ and $[\Box(P \wedge Q), \neg Q]$.

Disjunction:

$$\frac{\beta}{\beta_1,\beta_2}$$

Where β is any β-type formula (refer to chapter 2, section 3.3)

If there exists any occurrence of a β type formula in a clause we can replace the β formula by its corresponding β_1 and β_2 formulas. For example, if the β formula is $(\Box P \vee \neg Q)$ then its corresponding β_1 and β_2 formulas are $\Box P$ and $\neg Q$. Hence, if we have the clause: $[\Box P \vee \neg Q]$ then we can replace this clause with the clause $[\Box P, \neg Q]$.

Special Case Rule:

Unlike the rules above this rule operates on clause lists and uses the notion of a partition. We say P_1, P_2 is a partition of the clause C if P_1 and P_2 are disjoint and the members of P_1 and P_2 together are exactly the members of C. We define the special case rule as follows.

$$\frac{\left\langle \left[A_1,\ldots,A_n,B_1,\ldots,B_k\right],C_1,\ldots,C_m\right\rangle}{\left\langle \left[A_1,\ldots,A_n\right],C_1,\ldots,C_m\right\rangle,\left\langle \left[B_1,\ldots,B_k\right],C_1,\ldots,C_m\right\rangle}$$

Note that the partitions are $P_1 = [A_1, \ldots, A_n]$ and $P_2 = [B_1, \ldots, B_k]$ which gives $C = [A_1, \ldots, A_n, B_1, \ldots, B_k]$.

For example, if the special case rule is applied to the clause list ⟨[¬◊¬P, ¬◊¬Q], [¬□P], [¬□Q]⟩ then the two clause lists will be: ⟨[¬◊¬Q], [¬□P], [¬□Q]⟩ and ⟨[¬◊¬P], [¬□P], [¬□Q]⟩.

KT4-π Rule:

Although, Fitting (1990a) describes two rules for the system KT4 we will only be looking at the rule he calls KT4-π.

Let L be a clause list containing a clause C consisting entirely of π-formulas. Create a new clause C^{π}. For each π formula in C place its corresponding π_0 formula into C^{π}. Then L can be replaced by L* where L* is a clause list which consists of the clauses: C^{π}, and each clause C_i in L that consists entirely of ν formulas.

$$\frac{\left\langle \left[\pi_1,\ldots,\pi_n\right],\left[\nu_1,\ldots,\nu_m\right],\ldots,\left[\nu_1,\ldots,\nu_n\right],C_1,\ldots,C_k\right\rangle}{\left\langle \left[\pi_{0_1},\ldots,\pi_{0_n}\right],\left[\nu_1,\ldots,\nu_m\right],\ldots,\left[\nu_1,\ldots,\nu_n\right]\right\rangle}$$

For example, consider the following clause list: ⟨[¬□P, ◊Q], [□P, ¬◊Q], [□Q], [P, Q]⟩. Applying the KT4-π rule to this clause list results in the following clause list: ⟨[¬P, Q], [□P, ¬◊Q], [□Q]⟩.

KT4L Rule:

Fitting (1990a) does not provide a rule for the normal system of modal logic KT4L. However, the following rule, which we call the KT4L rule, is sound in his system.

Let L be a clause list containing clauses C_i for $i \geq 1$ consisting entirely of π-formulas. For each C_i create a new clause C_i^{π}. For each π formula in C_i place its corresponding π_0 formula into its corresponding clause C_i^{π}. For each C_i^{π} create a clause list L_i that contains the following:

- Each clause C in L that consists entirely of ν formulas.
- C_i^{π}
- Each clause C_j in L that consists entirely of π formulas such that $C_j \neq C_i$.

L can now be replace by L* where L* is the list of L_i.

$$\frac{\left\langle \left[\pi_1, \pi_2 \ldots, \pi_i\right], \ldots, \left[\pi_1, \pi_2, \ldots \pi_k\right], \left[\nu_1, \nu_2, \ldots, \nu_m\right], \ldots, \left[\nu_1, \nu_2, \ldots, \nu_n\right], C_1, C_2, \ldots C_q \right\rangle}{\left\langle \left[\pi_{0_1}, \pi_{0_2}, \ldots, \pi_{0_i}\right], \ldots, \left[\pi_1, \pi_2, \ldots, \pi_k\right], \left[\nu_1, \nu_2, \ldots, \nu_m\right], \ldots, \left[\nu_1, \nu_2, \ldots, \nu_n\right] \right\rangle}$$

$$\cdot$$

$$\cdot$$

$$\left\langle \left[\pi_1, \pi_2, \ldots, \pi_i\right], \ldots, \left[\pi_{0_1}, \pi_{0_2}, \ldots, \pi_{0_k}\right], \left[\nu_1, \nu_2, \ldots, \nu_m\right], \ldots, \left[\nu_1, \nu_2, \ldots, \vec{\nu}_n\right] \right\rangle .$$

For example, if the initial clause list is: ⟨[¬□P], [◊Q], [□R], [P, □Q], [□¬P]⟩ then this clause list is replaced by the following two clauses after the application of the KT4L rule: ⟨[¬P], [◊Q], [□R], [□¬P]⟩, ⟨[¬□P], [Q], [□R], [□¬P]⟩.

5.3.2 *Addition Rules*

The resolution rule is the first addition rule that is defined. This rule is defined for all the normal systems of modal logic, that is, a separate resolution rule is not required for each system.

Resolution Rule:

$$\frac{[Z, A_1, \ldots, A_k], [\neg Z, B_1, \ldots, B_k]}{[A_1, \ldots, A_k, B_1, \ldots, B_k], [Z, A_1, \ldots, A_k], [\neg Z, B_1, \ldots, B_k]}$$

where Z and ¬Z are complementary literals and the A_i's and B_i's are formulas.

For instance, if we have the two clauses in a clause list ⟨[¬P, □Q], [P, ◊R]⟩ then if we apply the resolution rule the clause list becomes: ⟨[□Q, ◊R], [¬P, □Q], [P, ◊R]⟩.

T-ν Rule:

The T-ν rule is an addition rule that is applicable for all reflexive models and is defined as follows:

If a clause list contains a clause C with an occurrence of a formula ν then another clause may be added to the clause list that is like C except that it contains occurrences of ν_0 where C has ν.

$$\frac{[\nu, A_1, \ldots, A_n]}{[\nu_0, A_1, \ldots, A_n], [\nu, A_1, \ldots, A_n]}$$

where the A_i's are formulas and ν, ν_0 are a ν-type formulas defined above.

For example, consider the ν-type formula □(P ∧ Q); its corresponding ν_0 formula is just (P ∧ Q). Hence if we have the clause ⟨[□(P ∧ Q), ¬P, R]⟩ then we will have ⟨[(P ∧ Q), ¬P, R], [□(P ∧ Q), ¬P, R]⟩ after the application of the T-ν rule.

5.3.3 *Summary*

The following table provides a summary of the rules that are applicable for the systems KT4 and KT4L.

System	Rules
KT4	• Double negation • Conjunction • Disjunction • Special case rule • T-ν rule • KT4-π rule • Resolution Rule
KT4L	• Double negation • Conjunction • Disjunction • Special case rule • T-ν rule • KT4L rule • Resolution Rule

5.4 Soundness and Completeness

5.4.1 *KT4*

Theorem 48 (KT4-π Soundness)

If A, some formula, is a theorem then A is valid in the class of reflexive and transitive standard models.

Proof: Only a proof sketch is provided here, the reader is referred to Fitting (1990a, 1983) for details. But first we require the following definition:

Definition

A block B is called ***satisfiable*** *if there is some reflexive and transitive model $M=<W, R, V>$ and some world $\alpha \in W$ such that B is true at α.*

To prove soundness we show that if we start with a satisfiable block then after applying the replacement and addition rules for the system KT4 we will still have a satisfiable block. From this we can conclude that if we derive a closed block from the initial block of clause lists then the initial block was not satisfiable. It follows therefore that if the empty block was derived from the initial block [⟨¬A⟩] then ¬A was not satisfiable and therefore A is valid in the class of reflexive transitive models. □

Theorem 49 (KT4-π Completeness)

If a formula A is valid in a class of reflexive and transitive models then A is a theorem.

Proof: We provide a brief sketch for this proof (the reader is referred to the work of Fitting 1990a & 1983 for the technical details). Similar to the proof of theorem 41 this completeness proof is based on a set of consistency properties and its corresponding model existence theorem. The consistency property and model existence theorem are defined as follows:

1. **Definition**

   We call a finite set {A_1, ..., A_n} of formulas ***consistent*** *if the block [⟨[A_1], ..., [A_n]⟩] is not closable.*

2. **Model Existence**

   The consistent set {A_1, ..., A_n} is satisfiable (Fitting 1990a:51).

For completeness: If A is not a theorem then by definition the block [⟨[¬A]⟩] is not closable. It follows from 1, that {¬A} is consistent. Then by 2, [⟨[¬A]⟩] is satisfiable and hence A is not valid in the class of reflexive and transitive models. □

5.4.2 *KT4L*

We do not prove a formal soundness and completeness result here due to the fact that the above KT4L rule is similar to that given by Goré (1995) in the next chapter and a formal proof can be found in his work.

Goré shows that his KT4 tableau system is both sound and complete with respect to the class of all reflexive and transitive standard models. In the previous section it was shown that KT4 is both sound and complete for destructive modal resolution. Now KT4L is merely an extension of KT4, that is, the only difference between the two systems is the addition of the rule KT4L and the removal of the rule KT4. Goré (1995) shows that this extension is sound and complete with respect to the class of all reflexive, transitive and weakly connected standard models. Since the KT4L rule above is identical to his KT4L rule (except for some notational alterations) we have that KT4L for destructive modal resolution is also sound and complete for the class of reflexive, transitive and weakly connected standard models.

5.5 Applications

5.5.1 *KT4*

In this section an application of the resolution system for KT4 is demonstrated. Assume that we want to show that formula $\Box(P \to Q) \to \Box(\Diamond P \to \Diamond Q)$ is valid in the class of all reflexive transitive models (i.e. we want to show that $\Box_{KT4}\Box(P \to Q) \to \Box(\Diamond P \to \Diamond Q)$. By definition 62 this means that we need to show that the block $[\langle[\neg(\Box(P \to Q) \to \Box(\Diamond P \to \Diamond Q))]\rangle]$ is closable. The proof is given below with the justification for the results on the left given on the right.

$[\langle[\neg(\Box(P \to Q) \to \Box(\Diamond P \to \Diamond Q))]\rangle]$	
$[\langle[\Box(P \to Q)], [\neg\Box(\Diamond P \to \Diamond Q)]\rangle]$	Conjunction rule
$[\langle[\Box(P \to Q)], [\neg(\Diamond P \to \Diamond Q)]\rangle]$	KT4-π rule
$[\langle[\Box(P \to Q)], [\Diamond P], [\neg\Diamond Q]\rangle]$	Conjunction rule
$[\langle[\Box(P \to Q)], [P], [\neg\Diamond Q]\rangle]$	KT4-π rule
$[\langle[P \to Q], [\neg Q], [\Box(P \to Q], [P], [\neg\Diamond Q]\rangle]$	T-ν rule applied twice
$[\langle[\neg P, Q], [\neg Q], [\Box(P \to Q)], [P], [\neg\Diamond Q]\rangle]$	Disjunction rule
$[\langle[\neg P], [\neg P, Q], [\neg Q], [\Box(P \to Q)], [P], [\neg\Diamond Q]\rangle]$	Resolution rule $[\neg P, Q], [\neg Q]$
$[\langle[], [\neg P], [\neg P, Q], [\neg Q], [\Box(P \to Q)], [P], [\neg\Diamond Q]\rangle]$	Resolution rule $[\neg P], [P]$

Since every clause list in the block contains the empty clause, the block is closed and it follows that the formula, $\Box(P \to Q) \to \Box\,(\Diamond P \to \Diamond Q)$ is valid in the class of all reflexive and transitive models.

5.5.2 *KT4L*

Similar to the example above we use definition 62 to show that the formula $\Box(P \wedge \Box P \rightarrow Q) \vee \Box(Q \wedge \Box Q \rightarrow P)$ is a theorem.

$[\langle[\neg(\Box(P \wedge \Box P \rightarrow Q) \vee \Box(Q \wedge \Box Q \rightarrow P))]\rangle]$	
$[\langle[\neg\Box(P \wedge \Box P \rightarrow Q)],[\neg\Box(Q \wedge \Box Q \rightarrow P)]\rangle]$	Conjunction Rule
$[\langle[\neg(P \wedge \Box P \rightarrow Q)], [\neg\Box(Q \wedge \Box Q \rightarrow P)]\rangle,$	KT4L Rule
$\langle [\neg\Box(P \wedge \Box P \rightarrow Q)],[\neg(Q \wedge \Box Q \rightarrow P)]\rangle]$	
$[\langle[P \wedge \Box P], [\neg Q], [\neg\Box(Q \wedge \Box Q \rightarrow P)]\rangle,$	Conjunction Rule applied twice.
$\langle [\neg\Box(P \wedge \Box P \rightarrow Q)], [Q \wedge \Box Q], [\neg P]\rangle]$	
$[\langle[P], [\Box P], [\neg Q], [\neg\Box(Q \wedge \Box Q \rightarrow P)]\rangle,$	Conjunction rule applied twice.
$\langle [\neg\Box(P \wedge \Box P \rightarrow Q)], [Q], [\Box Q], [\neg P]\rangle]$	
$[\langle[\Box P], [\neg(Q \wedge \Box Q \rightarrow P)]\rangle,$	KT4L rule applied twice.
$\langle[\neg(P \wedge \Box P \rightarrow Q)],[\Box Q]\rangle]$	
$[\langle[\Box P], [Q \wedge \Box Q], [\neg P]\rangle,$	Conjunction rule applied twice.
$\langle[P \wedge \Box P], [\neg Q], [\Box Q]\rangle]$	
$[\langle[\Box P], [Q], [\Box Q], [\neg P]\rangle,$	Conjunction rule applied twice
$\langle[P], [\Box P], [\neg Q], [\Box Q]\rangle]$	
$[\langle[\Box P], [P], [Q], [\Box Q], [\neg P]\rangle,$	T-∨ Rule ($\Box P$)
$\langle[P], [\Box P], [\neg Q], [\Box Q], [Q]\rangle]$	T-∨ Rule ($\Box Q$)
$[\langle[\Box P], [P], [Q], [\Box Q], [\neg P], []\rangle,$	Resolution Rule applied twice
$\langle[P], [\Box P], [\neg Q], [\Box Q], [Q], []\rangle]$	

From the above block we can see that both the clause lists are closed and hence the block is closed and therefore we can conclude that the initial formula is valid in the class of all reflexive, transitive and weakly connected models.

5.5.3 Special Case Rule

We include one more example in order to demonstrate the use of the special case rule. We show that the formula $\neg(\Box P \vee \Box Q) \vee (\neg\Diamond\neg P \vee \neg\Diamond\neg Q)$ is valid in the class of all reflexive and transitive standard models, that is, we will use the rules defined for the normal system of modal logic KT4.

$[\langle[\neg(\neg(\Box P \vee \Box Q) \vee (\neg\Diamond\neg P \vee \neg\Diamond\neg Q))]\rangle]$	
$[\langle[\neg\neg(\Box P \vee \Box Q)], [\neg(\neg\Diamond\neg P \vee \neg\Diamond\neg Q)]\rangle]$	Conjunction Rule
$[\langle[\Box P \vee \Box Q], [\neg(\neg\Diamond\neg P \vee \neg\Diamond\neg Q)]\rangle]$	Negation Rule
$[\langle[\Box P \vee \Box Q], [\neg\neg\Diamond\neg P], [\neg\neg\Diamond\neg Q]\rangle]$	Conjunction Rule
$[\langle[\Box P \vee \Box Q], [\Diamond\neg P], [\Diamond\neg Q]\rangle]$	Negation Rule applied twice
$[\langle[\Box P, \Box Q], [\Diamond\neg P], [\Diamond\neg Q]\rangle]$	Disjunction Rule
$[\langle[\Box P], [\Diamond\neg P], [\Diamond\neg Q]\rangle$ $\langle[\Box Q], [\Diamond\neg P], [\Diamond\neg Q]\rangle]$	Special Case Rule
$[\langle[\Box P], [\neg P]\rangle$ $\langle[\Box Q], [\neg Q]\rangle]$	KT4-π rule applied twice
$[\langle[\Box P], [P], [\neg P]\rangle$ $\langle[\Box Q], [Q], [\neg Q]\rangle]$	T-ν rule applied twice
$[\langle[\Box P], [], [P], [\neg P]\rangle$ $\langle[\Box Q], [], [Q], [\neg Q]\rangle]$	Resolution Rule applied twice

From the above block we can see that both the clause lists are closed and hence the block is closed and therefore, we can conclude that the initial formula is valid in the class of all reflexive, transitive standard models. Also note that without the use of the special case rule we would not have been able to close the above block.

5.6 Propositional Modal Logics KT5.

Fitting (1990a) notes in his article that destructive modal resolution is not applicable for modal logics whose models require symmetry. This means that it is not possible to apply destructive modal resolution directly to the modal logic KT5 since its models are symmetric. Alternatively it is possible to convert every formula in KT5 to a formula of modal degree one (i.e. a formula with no nested modal operators). An algorithm for this conversion can be found in Fitting (1983). Since the propositional logics KT5 and KT4 have the same valid formulas of degree one (see Fitting 1983) the resolution system for KT4 can therefore be applied to KT5.

6. CONCLUSION

If we survey the above resolution systems we will find that they can be divided into essentially two groups. Those that use translation-based methods and those that don't. In other words, those systems that convert the connective ◊ to some other connective and those that keep the connective as is and apply rules that manipulate the connective accordingly. Del Cerro & Herzig (1988) and Chan (1987) are examples of techniques that use translation based proofs. As shown above these techniques translate the modal connective ◊ into an alternative connective before applying the appropriate resolution rules. Unfortunately this translation still requires special rules in order to accommodate the new connective, that is, even though the translation simplifies the formulas into clauses, it is not always intuitive which clauses to resolve with one another. For instance in classical propositional resolution we replace P and ¬P with the empty clause. In the above methods we would have to either derive the clause ⊥ or run the clauses through a complex algorithm in order to determine whether the two clauses are unsatisfiable. It is interesting to note that both these techniques use clause forms, that is, similar to conventional resolution systems, they provide routines for converting formulas into clause form, choosing to apply their resolution rules to the clausal forms as opposed to the formulas. This reduction of formulas into clause form results in lists of clauses whose computational complexity is less than that of their initial formulas. Ophelders & De Swart (1993) argue that it is this reduction in computational complexity that is responsible for the success of most resolution-based theorem provers and not necessarily the use of resolution itself. It is not clear, however, whether this reduction in complexity would add to the efficiency of these systems were they to be automated.

Examples of resolution systems that don't use translation based methods are those of Fitting (1990b) and Abadi & Manna (1986). Once again it is interesting to note that both these techniques do not make use of clausal forms but chose instead to convert the formulas while applying the appropriate rules. Although Fitting (1990b) introduced new notation for the application of resolution (e.g. clause lists, blocks) his resolution rule appeared to be the most intuitive or closely related to the conventional application than any of the other approaches. In other words, his rules resolved complementary *literals* with each other as opposed to m-literals or other modal variations of complementary literals.

All the resolution systems introduced special rules for the normal systems of modal logic. They were all able to show soundness and completeness for at least S4. Abadi & Manna provided soundness and completeness for KT5 and Fitting (1990b) and del Cerro & Herzig (1988) noted that their systems could be extended to KT5.

Very little can be said about the efficiency of these systems since no implementation results where provided with the description of these systems. Chan (1987) did provide an implementation of his pattern matching algorithm in Franz LISP but no conclusive results where provided with regards to the overall efficiency of this algorithm. The discerning reader may also have noted that some of resolution proof systems described above make use of non deterministic rules for instance, Abadi & Manna's (1986) resolution rule and Fitting's (1990b) special case rule. The application of these rules, if implemented, will result in search spaces that require a great deal of backtracking and therefore result in proof procedures that may not perform optimally. However, as Ian Gent (1993) noted in his thesis, there is very little information available with regard to the implementation resolution proof systems, and therefore, we can merely speculate when it comes to the efficiency of the above techniques.

Conclusive results can only be derived once these resolution systems have been implemented and their applications towards resolving reasoning problems in modal logics have been efficiently demonstrated.

Resolution proof systems are not the only proof systems available for the determination of local and global logical entailment in propositional modal logic. Tableau proof systems, as introduced in chapter 2 (see section 3.3), have also been used with much success in propositional modal logic. The next chapter will survey these proof systems providing examples of applications of these systems for the normal systems of modal logic KT4, KT4L and KT5.

CHAPTER 5

TABLEAUX SYSTEMS WITHIN MODAL LOGIC

Just because genuine science would be impossible without an ordered and regular world which humans can to some extent get to know, does not prove that science is properly grounded. Its much vaunted success may be illusory.

- R Trigg.

1. INTRODUCTION

In chapter 2 we noted that tableau systems are essentially refutation systems that decompose a given set of formulas into a network of formulas which can be represented by trees. In this chapter we extend this notion to modal logic, in such a way that the trees of formulas now also incorporate possible worlds in the associated modal models. Similar to the previous chapter we have selected various articles that represent the current work in modal logic with regards to tableau systems. We chose Melvin Fitting (1983) in particular because his work is well known, provides a good introduction to tableau systems within modal logic and is often referred to by other authors of modal tableaux systems. Rajeev Goré (1995) has probably provided the most current and complete analysis of modal tableau systems and we therefore provide an overview of his tableau systems here. We recommend strongly that the interested reader refer to his report for a complete exposition on his work.

The work of the authors will be presented in terms of the following headings:

- Overview - a brief description of the system
- Syntax and semantics - definitions and descriptions of terminology not defined before.
- Rules - applicable rules for each system
- Soundness and completeness
- Applications - examples of how to apply the different tableau systems
- Decidability results.

Similar to chapter 4, chapter 5 is by no means a formal expose on the tableau systems introduced here but merely a description on how they work. For the sake of conformity to previous chapters notational alterations have been made to the different tableau systems.

Chapter 5 is concluded with a brief comparison of the two approaches and some comments on their efficiency.

2. TABLEAU SYTEMS - MELVIN FITTING

Fitting's (1983) tableau systems are similar to the tableau systems described in chapter 2, with the exception that over and above the propositional classical tableau rules the nodes in the tableaux have modal formulas as well. We will only review that section of his work that covers the propositional modal systems KT4 and KT5.

2.1 Syntax and Semantics

For the sake of convenience Fitting (1983) describes his formulas in terms of types. In the propositional case he describes two types of formulas namely α and β formulas. These formulas are exactly those which we encountered in chapter 2. In the modal case he extends his sets of formulas to also include ν and π formula types. We encountered Fitting's (1983) ν and π formulas when we discussed Destructive Modal Resolution in chapter 4, and will therefore not repeat them here. For the interested reader the majority of Fitting's (1983) syntax is based on the ideas of Kripke (1959, 1963).

In the following discussion L is KT4, KT4L or KT5. Note that for the sake of convenience we will make use of the terms L-model, L satisfiable and L tableau system. By L-model we mean the class of standard models corresponding to L. By L satisfiable we mean (see definition 21) that the L-model M=<W, R, V> satisfies some formula A; that is, for some $\alpha \in$ W, $\square_{\alpha}^{M}$ A. By a L tableau system we mean a tableau system for a particular modal logic L.

As mentioned in chapter 2 (see definition 16) a tableau proof of a formula A from a set of formulas Γ, is a closed tableau for $\Gamma \cup \{\neg A\}$. Fitting (1983), however, distinguishes between two types of tableau proofs. One in which we only deal with a single formula A, that is, a *tableau proof* for a formula A is just a closed tableau for $\{\neg A\}$. Effectively we are showing $\square_{C}$ A where $\square$ is local logical entailment, C the class of standard models corresponding to L and Γ is the empty set. However, it is not always possible to only work with one formula at a time, sometimes we require several sets of formulas. A tableau using this variation is described by means of the following definition (Fitting 1983:70):

Definition 63

Let Γ, Γ_L and Γ_G be arbitrary sets of formulas. By an ***L tableau*** *for Γ using members Γ_L as local assumptions and members of Γ_G as global assumptions (denoted $\Gamma_G \Box \Gamma_L \to \Gamma$) we mean any tableau that:*

- *Begins by putting down a finite subset of Γ (Γ may be infinite) in any order.*
- *Proceeds according to the usual tableau rules for L.*
- *But allows the following two assumption rules:*

1. ***Local assumption rule:-*** *Before any of the usual tableau rules for L are applied, any member of Γ_L may be added to the end of the branch.*

2. ***Global assumption rule****:- At any point in the tableau construction, any member of Γ_G may be added to the end of any branch.*

In effect this definition says that for every model M=<W, R, V> in the class of standard models corresponding to L, in which all the formulas of Γ_G are true $\Gamma_L \to \Gamma$ must also be true in M. Now $\Gamma_L \to \Gamma$ means that if all the members of Γ_L are true at a world $\alpha \in W$ then at least one member of Γ is also true at α. In other words $\Gamma_G \Box \Gamma_L \to \Gamma$ is a combination of global and local logical entailment as specified in chapter 3, where $\Box$ denotes global logical entailment and $\to$ denotes local logical entailment.

A proof therefore, for a set of formulas Γ, using the set of formulas Γ_G as global assumption and Γ_L as local assumptions, would be a closed tableau for the set $\neg\Gamma = \{\neg A \mid A \in \Gamma\}$, that is, in order to conclude $\Gamma_G \Box \Gamma_L \to \Gamma$ we need to find a closed tableau for $\neg\Gamma$.

2.2 Tableau Systems For KT4

2.2.1 *Rules*

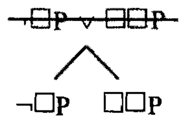
Essentially there are two types of rules in the tableau systems defined by Fitting (1983). The first set of rules could be considered as 'expansion' rules, that is, when the tableau rules are applied the current tableau is effectively expanded by new nodes that contain new formulas. The second type is the set of 'modification' rules. As with the 'expansion' rules these rules also expand the tableau by new nodes but at the same time these rules modify the current branch by deleting formulas that are no longer applicable. For the purpose of this thesis a formula that has been deleted will be denoted by a formula that has been crossed out, e.g. ~~□A ∧ □□A.~~

Note that only formulas that have not been deleted are available for the remainder of the proof, that is, if $\neg A$ and A occur on the same branch then we cannot conclude that the branch is closed since $\neg A$ has been deleted.

The following rules are applicable to the tableau system KT4.

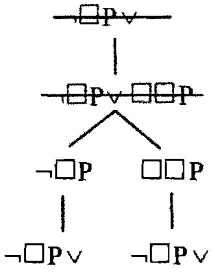
- All the tableau expansion rules for the propositional case (refer to chapter 2)

  These rules are modification rules as discussed above, that is, once the propositional rules have been applied to the current node (the node we are currently working with), the current node is deleted. For instance if the node on the tableau contains the formula $\neg\Box P \vee \Box\Box P$ then the tableau can be expanded by means of the β rule (see tableau expansion rules in chapter 2) to create two new branches with nodes $\neg\Box P$ and $\Box\Box P$ respectively. Illustrated by:



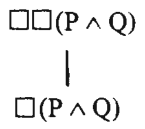
- The repetition rule.

  This rule is applicable when a number of branches share a common set of formulas. In such a case the common formulas can be copied to each separate branch, deleting the original formula, so that the branches can be updated separately without affecting the other branches which share the same formulas (Fitting 1983). In the example below the formula $\neg\Box P \vee P$ has been copied to each branch after the application of the β rule to the formula $\neg\Box P \vee \Box\Box P$.



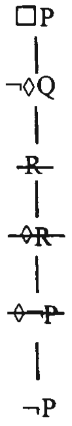
- The ν rule: $\frac{\nu}{\nu_0}$ where ν and ν_0 are as defined previously.

The ν rule is an instance of an expansion rule, that is, no modifications are required to the tableau other than the addition of a new node. For example, if the current node contains the formula □□(P ∧ Q) the tableau can be expanded by means of the ν rule to create the new node containing the formula □(P ∧ Q). Illustrated by:

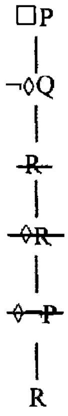


- The π rule: $\frac{\Gamma, \pi}{\{\nu | \nu \in \Gamma\}, \pi_0}$ where Γ is the set of formulas currently on the branch and π, π_0 and ν is as was previously defined.

In other words, let Γ be the set of formulas that are currently on the branch and let π be some formula on the same branch as Γ that is not in the set of formulas Γ. For instance in the tableau below Γ = {□P, ¬◊Q, R, ◊R} and π = ◊¬P. After the application of the π rule Γ is replaced by all the ν formulas in Γ, i.e. the set {□P, ¬◊Q} and π is replaced by ¬P. Implementing this result on the tableau involves deleting the formulas R, ◊¬P and ◊R and extending the tableau by an additional node which contains ¬P. This is illustrated as follows:



Note that we could also have used π = ◊R. In other words the π rule is not deterministic. Therefore it is possible that when we get to the end of the tableau that we may have to backtrack if we have not been successful in finding a proof. By backtracking we mean that we will have to remove all the branches of the tableau until we reach the node where we discarded a π formula and restart the tableau from this point. So in the example above we would remove ¬P and apply the π rule using π = ◊R, which would result in the following tree.



There are a number of alternative ways to deal with this issue of non-determinism. Another alternative would be to create a separate tableau for each application of the π rule. For instance, in the above example we would be working with two different tableaux in parallel after the application of the π rule. If either of the tableaux became closed we would have a proof. In other words both tableaux do not need to be closed. The same holds true for the generic case, that is, if we are working with n tableaux than only one has to close. This issue of non-determinism will not be raised in the remainder of this work but the reader should be aware that this issue will have to be addressed if this system was to be implemented on a computer system.

2.2.2 Soundness and Completeness

Theorem 50 (Soundness)

For any formula A, if A has a tableau proof (see section 2.1 above) using the respective KT4 rules, then A is valid in the class of all reflexive transitive models.

Proof: The proof is very similar to the one given for soundness in chapter 4 under destructive modal resolution, that is, if the proof is started with a satisfiable set of formulas then after the application of the respective rules for the KT4 system the set of formulas will still be satisfiable. From this it follows that if the closed tableau (see chapter 2) is derived from the formula ¬A then ¬A is not satisfiable in the class of all reflexive transitive models and therefore it follows that A is valid. A complete proof is available in Fitting (1983:46). □

To show completeness Fitting (1983) essentially uses a Lindenbaum (see chapter 3 section 6.3) type construction, that is, he defines a consistent set which he extends to a maximal consistent one. He then uses the maximal consistent set to construct a model for his tableau systems.

Theorem 51 (Completeness)

If a formula A is valid in all the class of all reflexive transitive models, then A has a tableau proof using the respective KT4 rules.

Proof: The proof follows a contrapositive argument, that is, we assume there does not exist a proof for a formula A from a KT4 tableau system and then proceed to show that A is not valid in the class of standard models corresponding to KT4. A complete proof can be obtained in Fitting (1983:61-63). □

We state the theorems for strong soundness and completeness without proof :

Theorem *52 (Strong Soundness)*

Let Γ, Γ_G and Γ_L be sets of formulas. If there is a closed tableau for $\neg\Gamma$ using the members of Γ_G as global assumptions and the members of Γ_L as local assumptions then $\Gamma_{G\Box}\ \Gamma_L \rightarrow \Gamma$.

(See Fitting 1983:72 for details of proof.)

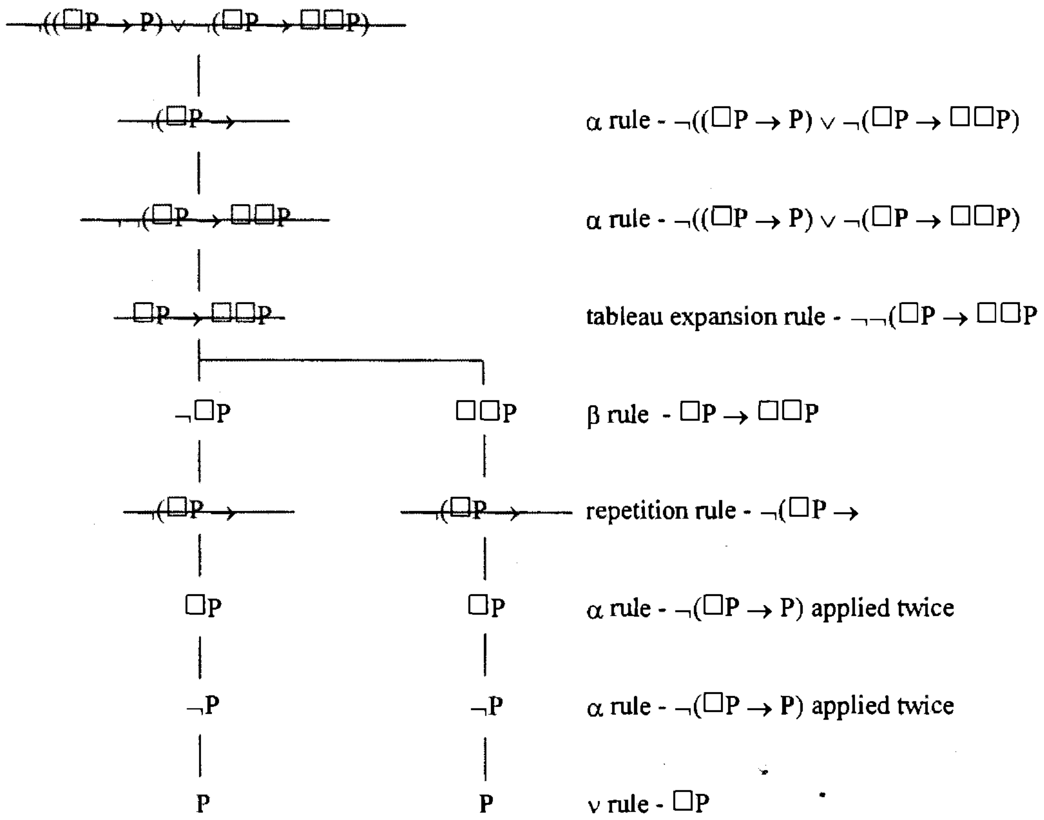
Theorem *53 (Strong Completeness)*

Let Γ, Γ_G and Γ_L be sets of formulas. If $\Gamma_{G\Box}\ \Gamma_L \rightarrow \Gamma$ then there is a closed tableau for $\neg\Gamma$ using the members of Γ_G as global assumptions and the members of Γ_L as local assumptions.

(See Fitting 1983:73 for details of proof.)

2.2.3 *Applications*

In this section we show a tableau proof for a single formula. (A tableau proof that makes use of local and global assumptions is demonstrated in the section 2.3.4 below.) Assume that we want to show that $(\Box P \to P) \vee \neg(\Box P \to \Box\Box P)$ is valid in the class of all reflexive transitive models. Then by soundness we know that if there is a tableau proof for $\neg((\Box P \to P) \vee \neg(\Box P \to \Box\Box P))$ then $(\Box P \to P) \vee \neg(\Box P \to \Box\Box P)$ is valid with respect to this class. Negating $(\Box P \to P) \vee \neg(\Box P \to \Box\Box P)$ the proof follows.



~~¬((□P → P) ∨ ¬(□P → □□P))~~
~~¬(□P → P)~~
α rule - ¬((□P → P) ∨ ¬(□P → □□P)
~~¬¬(□P → □□P~~
α rule - ¬((□P → P) ∨ ¬(□P → □□P)
~~□P → □□P~~
tableau expansion rule - ¬¬(□P → □□P
¬□P
□□P
β rule - □P → □□P
~~¬(□P → P)~~
~~¬(□P → P)~~
repetition rule - ¬(□P →
□P
□P
α rule - ¬(□P → P) applied twice
¬P
¬P
α rule - ¬(□P → P) applied twice
P
P
ν rule - □P

Since both branches are closed (both contain P and $\neg P$) we have a closed tableau and it follows that $(\Box P \to P) \vee \neg(\Box P \to \Box\Box P)$ is valid in the class of all reflexive transitive models. Note that the left most column is not part of the proof but merely an indication to the reader as to what rules where used to obtain the required nodes. So for instance nodes $\neg(\Box P \to P)$ and $\neg\neg(\Box P \to \Box\Box P)$ were obtained after the application of the α rule to the formula $\neg((\Box P \to P) \vee \neg(\Box P \to \Box\Box P))$.

2.2.4 *Decidability*

In this section we will show that the KT4 tableau system, as defined by Fitting(1983), is decidable.

First the notion of KT4-consistency property is defined. The consistency property was initially introduced in chapter 4 (see section 2.5) and can be defined for the KT4 system as follows (Fitting 1983:50).

Definition 64

Let C be a collection of sets of formulas . C is a ***KT4 consistency property*** *if it meets the following conditions: For each $\Gamma \in C$:*

- *Γ does not contain A and $\neg A$ for any literal A..*
- *Γ does not contain $\bot$ or $\neg T$.*
- *if $\alpha \in \Gamma$ then $\Gamma \cup \{\alpha_1, \alpha_2\} \in C$*
- *if $\beta \in \Gamma$ then $\Gamma \cup \{\beta_1\} \in C$ or $\Gamma \cup \{\beta_2\} \in C$*
- *if $\nu \in \Gamma$ then $\Gamma \cup \{\nu_0\} \in C$*
- *if $\pi \in \Gamma$ then $\{\nu \mid \nu \in \Gamma\} \cup \{\pi_0\} \in C$*

The reader will recall that subformulas were defined in chapter 3 (definition 35) and that it was noted that a set of formulas Γ is closed if and only if Γ contained all its subformulas. This notion in conjunction with the KT4 consistency property lends itself to the following definition.

Definition 65

Let C be a set of formulas which is a KT4 consistency property and let Γ be an arbitrary collection of formulas. By C **restricted** *to Γ, we mean the collection of all sets of the form $S \cap \Gamma$ where $S \in C$ (Fitting 1983:115).*

Theorem ***54***

Let A be a formula, C a KT4 consistency property and let $\Gamma(A)$ be the set of all subformulas of A. A is valid in all KT4 models if and only if $\{\neg A\}$ is not a member of any KT4 consistency property of the form C restricted to $\Gamma(A)$.

(See Fitting, 1983:116 for a proof.)

Let A be any formula, with n subformulas, $\Gamma(A)$ the set of all subformulas of A, C a KT4 consistency property. In order to show that A is KT4-valid (i.e. valid in all classes of standard models that correspond to KT4) we need to check, by theorem 54, every KT4 consistency property of the form C restricted to $\Gamma(A)$ to see if $\{\neg A\}$ is a member. Now if $S \in$ (C restricted to $\Gamma(A)$) then $S \subseteq \Gamma(A)$ and hence $S \in P(\Gamma(A))$ where P is the power set operation, that is, (C restricted to $\Gamma(A)$) $\subseteq P(\Gamma(A))$. Now $\Gamma(A)$ has at most 2n members, and therefore any KT4-consistency property of the form (C restricted to $\Gamma(A)$) has at most 2^{2n} members.

Since (C restricted to $\Gamma(A)$) $\subseteq P(\Gamma(A))$ we have that (C restricted to $\Gamma(A)$) $\in P(P(\Gamma(A)))$. It follows therefore that there are at most $2^{2^{2n}}$ KT4 consistency properties of the form (C restricted to $\Gamma(A)$). From this result we note that in order to show that A is KT4 valid we only need to verify a finite well determined number of sets, and hence it follows that there is a decision procedure for KT4. (Fitting, 1983:116,117.)

2.3 Tableau Systems for KT5

2.3.1 *Overview*

Although Fitting (1983) describes a number of tableaux systems for KT5 we will only be looking at his semi analytic KT5 tableau system, that is, his KT5 tableau system that makes use of the semi analytic cut rule (see definition below). Also, note that for the sake of convenience we will only be looking at strong soundness and completeness, that is, all the tableaux in this section will have been built as described by definition 63 . In other words given any set of formulas Γ (possibly infinite), Γ_G as global assumptions and Γ_L as local assumptions we can deduce Γ_G □ $\Gamma_L \rightarrow \Gamma$ iff we can find a closed tableau for $\neg\Gamma = \{\neg A \mid A \in \Gamma\}$.

2.3.2 *Rules*

Semi-Analytic Cut Rule

Before defining the relevant rules for KT5 we digress to discuss a rule known as the cut rule.

The cut rule essentially says that at any time during the construction of a tableau we can split the tableau into two branches and add A to the one branch and $\neg A$ to the other branch for any formula A.

We can denote this as follows: $\dfrac{}{A|\neg A}$ where A is any arbitrary formula.

The cut rule however has some drawbacks, one of them being that it is not possible to show decidability when using the cut rule. The cut rule adds arbitrary formulas to the tableau system therefore there is no guarantee that the tableau will eventually terminate. It is therefore not surprising that a number of researchers have tried to find alternative options to the cut rule. One of these researchers is Osamu Sonobe who defined a cut rule which is restricted to subformulas of formulas already on the branch and formulas that can be built up from these formulas by prefixing modal operators to the formulas. This restriction is defined in terms of a cut class as follows (Fitting 1983:202, 203):

Definition 66

Let Γ be a set of formulas. We call Γ_c a ***cut class*** *if:*

1. *Γ_c is closed under subformulas*
2. *Γ_c is closed under the operation of prefixing modal operators; that is, if A or $\neg A$ belongs to Γ_c then so do $\Box A$, $\Diamond A$, $\neg\Box A$ and $\neg\Diamond A$.*

Definition 67

Let Γ_c be a cut class. The cut rule ***restricted to Γ_c*** *is the rule: at any point in a tableau we may split the end of a branch, and add A to the one fork and $\neg A$ to the other, for any $A \in \Gamma_c$. An instance of the cut rule, restricted to Γ_c, is called a Γ_c*-***cut***, *denoted by: $\dfrac{}{A \mid \neg A}$ where $A \in \Gamma_c$.*

Now we can further restrict this cut rule to the smallest cut class as follows:

Definition 68

Let Γ be a set of formulas and A any formula. By $C(\Gamma)$ we mean the smallest cut class extending Γ and by $C(A)$ we mean $C(\{A\})$. We call cuts restricted to $C(\Gamma)$ and $C(A)$ ***semi analytical cuts***.

Unlike the ordinary cut rule, semi analytic cuts have the advantage that they place a check on the total arbitrariness inherent in the cut rule.

Summary Of Rules

The following rules have been defined for the tableau system KT5.

- All the propositional tableau expansion rules.

These rules are applied in a similar fashion to that described for KT4.

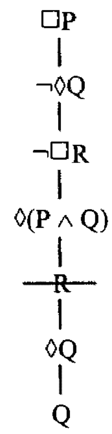
- The repetition rule, as described above.

- v-rule: $\dfrac{\nu}{\nu_0}$ where ν and ν_0 are as defined previously.

- π-rule: $\dfrac{\Gamma, \pi}{\{\nu | \nu \in \Gamma\} \cup \{\pi | \pi \in \Gamma\}, \pi_0}$

  where Γ is a set of formulas and ν, π and π_0 are as defined previously

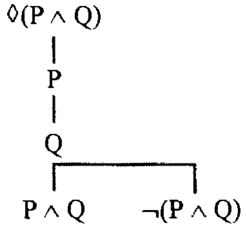
As with the π rule for KT4 this π rule is a replacement rule. Γ contains all the formulas that have not been deleted on the branch that is currently being worked on. π is also a formula on the branch but is not contained in Γ. After the application of the π rule the formulas on the branch are replaced by the sets $\{\nu | \nu \in \Gamma\} \cup \{\pi \mid \pi \in \Gamma\}$ and the corresponding π_0 formula of π. It follows therefore that all the formulas on the branch that are not in the set of all ν's or the set of all π's must be deleted. For example, in the tableau below $\Gamma = \{\Box P, \neg\Diamond Q, \neg\Box R, \Diamond(P \wedge Q), R\}$ and $\pi = \Diamond Q$. After the application of the π-rule Γ is replaced by the sets $\{\Box P, \neg\Diamond Q\} \cup \{\neg\Box R; \Diamond(P \wedge Q), \Diamond Q\}$ and π is replaced by $\pi_0 = Q$.



- Semi analytic cut rule: $\frac{}{A \mid \neg A}$

  where A is a formula restricted to the cut class $C(\Gamma \cup \Gamma_L \cup \Gamma_G)$. The cut class applicable to this rule is defined in terms of local and global assumptions as follows. Let Γ denote a set of formulas, Γ_L the set of local assumptions and Γ_G the set of global assumptions. Then the cut class for the semi analytical cut rule is of the type $C(\Gamma \cup \Gamma_L \cup \Gamma_G)$.

For example, if $\Gamma = \{\Diamond(P \wedge Q)\}$, $\Gamma_L = \{P, Q\}$ and Γ_G is the empty set, then the cut class is $C(\Gamma \cup \Gamma_L)$. Since C is a cut class we also know that by definition C is closed under subformulas and therefore $(P \wedge Q)$ also belongs to C. It follows therefore that if $A = (P \wedge Q)$ then we have the following application of the semi analytic cut rule:



2.3.3 *Soundness and Completeness*

Theorem *55*

Let Γ, Γ_L and Γ_G be sets of formulas. There is a closed semi analytic KT5 tableau for $\neg\Gamma = \{\neg A \mid A \in \Gamma\}$ using the members of Γ_L as local and the members of Γ_G as global assumptions if and only if Γ_G□ $\Gamma_L \rightarrow \Gamma$.

Proof: Soundness for KT5 follows a similar argument to soundness of KT4 (see theorem 50). Completeness follows a contrapositive argument similar to that described by Abadi & Manna (1986) in chapter 4, section 2.5. In this instance the consistency property and model existence theorem are defined as follows:

1. **Definition.**

Let C be a collection of sets of formulas. C is a ***classical consistency property*** *if it meets the following conditions. (Fitting 1983:48):*

1. *S does not contain A and* $\neg A$ *for any literal A..*
2. *S does not contain* $\perp$ *or* $\neg T$.
4. *if* $\alpha \in S$ *then* $S \cup \{\alpha_1, \alpha_2\} \in C$.
5. *if* $\beta \in S$ *then* $S \cup \{\beta_1\} \in C$ *or* $S \cup \{\beta_2\} \in C$.

If $S \in C$, *S will be referred to as* ***C-consistent.***

2. **Definition.**

Let Γ_c *be any cut class (see definition 66) . If C is a classical consistency property then we call C an* ***KT5-consistency property with cut class*** Γ_c *iff for each* $S \in C$:

1. *Cut condition: for any formula* $A \in \Gamma_c$ *either* $S \cup \{A\} \in C$ *or* $S \cup \{\neg A\} \in C$, *and*
2. *if* $\pi \in S$ *then* $\{\nu \mid \nu \in S\} \cup \{\pi \mid \pi \in S\} \cup \pi_0 \in C$. *(Fitting 1983:205.)*

3. **Definition**

Let C be a KT5-consistency property and let Γ *be a set of formulas. We call C* ***Γ-compatible*** *if for each* $S \in C$ *and for each* $A \in \Gamma$, $S \cup \{A\} \in C$ *(Fitting 1983:51).*

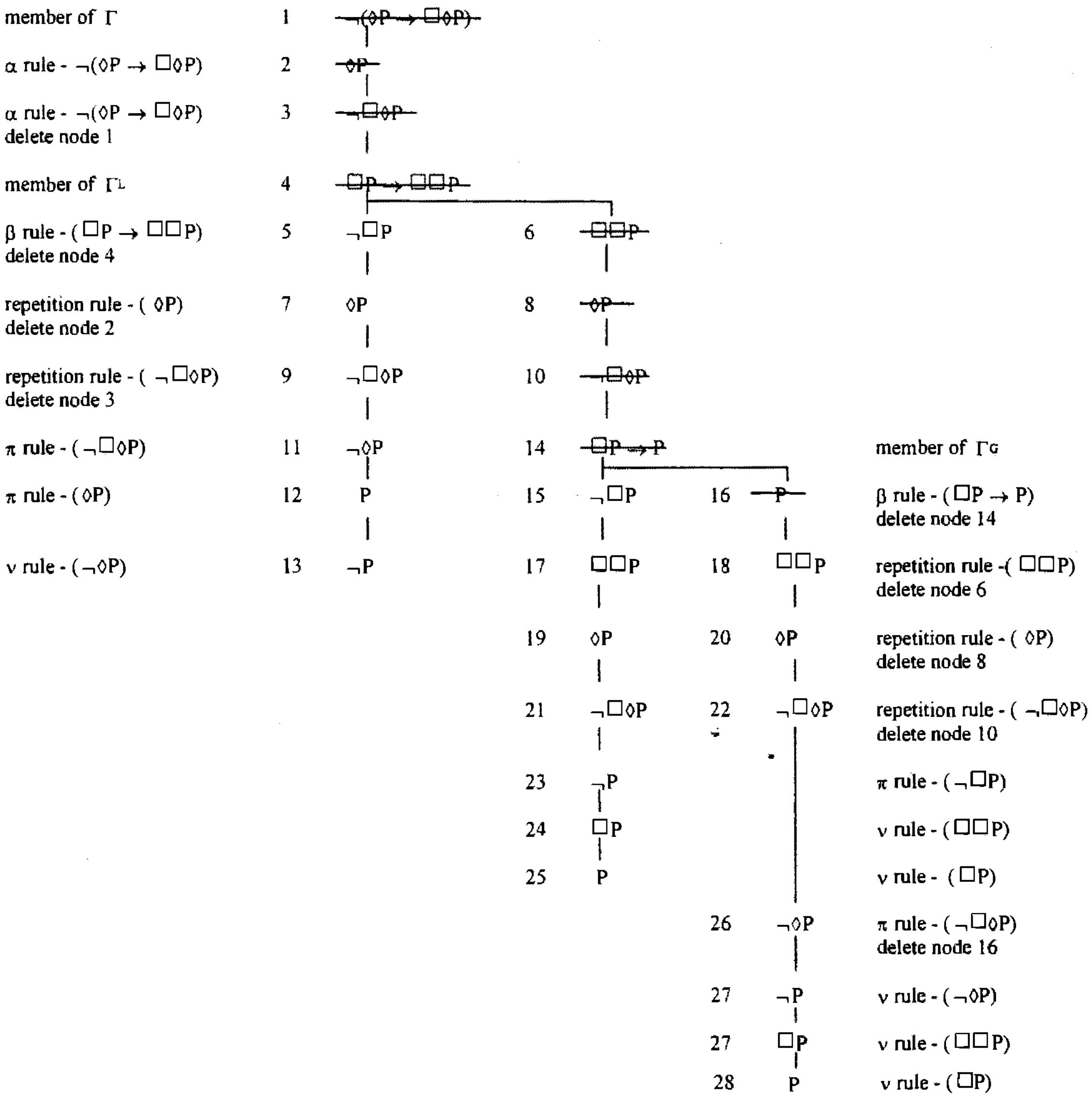
4. **Strong Model Existence**

Let Γ, Γ_L *and* Γ_G *be arbitrary sets of formulas. Assume that there is a KT5-tableau for* Γ *using the members of* Γ_L *as local assumptions and members of* Γ_G *as global assumptions. Let C be a KT5-consistency property with cut class* $\Gamma_c = C(\Gamma \cup \Gamma_G \cup \Gamma_L)$ *that is* Γ_L*-compatible. If* Γ_c *extends* Γ_G *and* Γ_L *then, if* Γ_G *is C-consistent,* Γ_G *is satisfiable at a world in some reflexive transitive symmetric model M=<W, R, V> in which the members of* Γ_L *hold at every possible world.* (See Fitting 1993:206 for proof.)

We refer the reader to Fitting (1993:209) for the details. □

2.3.4 Applications

The tableau system can be demonstrated for KT5 as follows. Let $\Gamma_G = \{\Box P \to P\}$, $\Gamma_L = \{\Box P \to \Box\Box P\}$ and $\Gamma = \{\Diamond P \to \Box\Diamond P\}$. We want to show $\Gamma_{G\Box}\, \Gamma_L \to \Gamma$. This means by theorem 55 that we have to show that there is a closed tableau for $\neg\Gamma$, i.e. $\neg(\Diamond P \to \Box\Diamond P)$. The proof follows below:



member of Γ	1	~~$\neg(\Diamond P \to \Box\Diamond P)$~~					
α rule - $\neg(\Diamond P \to \Box\Diamond P)$	2	~~$\Diamond P$~~					
α rule - $\neg(\Diamond P \to \Box\Diamond P)$ delete node 1	3	~~$\neg\Box\Diamond P$~~					
member of Γ_L	4	~~$\Box P \to \Box\Box P$~~					
β rule - $(\Box P \to \Box\Box P)$ delete node 4	5	$\neg\Box P$	6	~~$\Box\Box P$~~			
repetition rule - $(\Diamond P)$ delete node 2	7	$\Diamond P$	8	~~$\Diamond P$~~			
repetition rule - $(\neg\Box\Diamond P)$ delete node 3	9	$\neg\Box\Diamond P$	10	~~$\neg\Box\Diamond P$~~			
π rule - $(\neg\Box\Diamond P)$	11	$\neg\Diamond P$	14	~~$\Box P \to P$~~			member of Γ_G
π rule - $(\Diamond P)$	12	P	15	$\neg\Box P$	16	~~P~~	β rule - $(\Box P \to P)$ delete node 14
ν rule - $(\neg\Diamond P)$	13	$\neg P$	17	$\Box\Box P$	18	$\Box\Box P$	repetition rule -$(\Box\Box P)$ delete node 6
			19	$\Diamond P$	20	$\Diamond P$	repetition rule - $(\Diamond P)$ delete node 8
			21	$\neg\Box\Diamond P$	22	$\neg\Box\Diamond P$	repetition rule - $(\neg\Box\Diamond P)$ delete node 10
			23	$\neg P$			π rule - $(\neg\Box P)$
			24	$\Box P$			ν rule - $(\Box\Box P)$
			25	P			ν rule - $(\Box P)$
					26	$\neg\Diamond P$	π rule - $(\neg\Box\Diamond P)$ delete node 16
					27	$\neg P$	ν rule - $(\neg\Diamond P)$
					27	$\Box P$	ν rule - $(\Box\Box P)$
					28	P	ν rule - $(\Box P)$

Since each branch is closed, the tableau is closed and we have the desired result. Note that the left and right most columns as well as the node numbers are not part of the proof but merely an indication of how the nodes were derived. The reader may also notice that there was more than one way to show this proof. In other words, the proof to show $\Gamma_{G\Box}\Gamma_L \to \Gamma$ is not unique.

2.3.5 *Decidability*

Decidability results are not possible for tableau systems that use variations of the cut rule, such as the KT5 tableau system. However, Fitting (1983) does make some interesting observations with regards to decidability for the KT5 tableau system.

Fitting (1983) demonstrates that it is possible to show that a formula is valid in the class of models corresponding to KT5 (i.e. KT5 valid) without using the cut rule. This is done by allowing the addition of another rule that stipulates that from π_0 we can infer π. In other words, adding this rule to the KT5 tableau system means that we can dispense with the cut rule. Fitting calls this tableau system the weak KT5 tableau system. The weak KT5 tableau system is exactly the same as the KT5 tableau system discussed above except that the weak KT5 tableau system contains the rule $\frac{\pi_0}{\pi}$ instead of the semi analytical cut rule: $\frac{}{A \mid \neg A}$. However, Fitting (1983:226) is only able to show weak completeness for this tableau system and notes that strong completeness (where the sets of formulas may be infinite) is still an issue open to debate.

Although we will not discuss this system in detail here (the reader is referred Fitting 1983) we will note the following interesting results which provide the basis for the proof of weak completeness of a cut-free KT5 tableau system.

Theorem 56

A formula A, is KT5 valid if and only if $\Diamond\Box A$ is KT4 valid.

(See Fitting 1983:222-225 for details of proof.)

Theorem 57

KT5 is decidable.

Proof: Since KT4 is decidable. (Fitting 1983:222).

3. PREFIXED TABLEAU SYSTEMS - MELVIN FITTING

3.1 Syntax and Semantics

In the previous sections we discussed tableaux whose rules resulted in shifts from one world to another. For instance when we applied the π rule for KT4, to a branch B, in a particular tableau, we had to update all the formulas on the branch so that the branch reflected the current world that we had shifted to, that is, in these tableaux we always moved forward to the next world and erased the results from the past worlds. In symmetric models we have the option to return to past worlds, and therefore these tableaux are not always ideal. For these systems prefixed tableau systems, as described below, are far superior. Essentially prefixed tableau systems are tableau systems that make use of prefixed formulas where the prefixes are labels for the worlds at which the formulas are true in a given model, that is, the prefixes are used to keep track of the different worlds on the branch. It is therefore not necessary to delete formulas every time a rule is applied to the branch since the prefixes provide an explicit indication of the worlds.

Definition 69

*A **prefix** is a finite sequence of positive integers. A **prefixed formula** ℓA is a prefix ℓ followed by a formula A (Fitting 1983:388).*

For example, if the formula 1A appears on the branch of a tableau then it should be clear that A is a formula that is true at a world labelled by 1.

As with the tableau systems discussed previously we also start a tableau proof for a formula ℓA by $\ell\neg$A. Which essentially means that there is a world labelled by ℓ at which $\neg$A is true. For the sake of convenience we will redefine closure in terms of prefixed tableau systems.

Definition 70

*A tableau branch is **closed** if it contains ℓA and $\ell\neg A$ for some prefix ℓ and some formula A. A tableau is closed if every branch is closed. A closed tableau for $\ell\neg A$ is a **proof** of A (Fitting 1983:389).*

It follows therefore that if we find a closed table for $\ell\neg$A we can conclude $\square_C$A where $\square$ is local logical entailment and C the class of standard models corresponding to L.

In order to apply the rules that are defined in the next section we require the following terminology with regard to prefixes.

Definition 71

We say a prefix ℓ is ***used*** *on a tableau branch if ℓA occurs on the branch for some formula A. We say prefix ℓ is* ***unrestricted*** *on a tableau branch if ℓ is not an initial segment (proper or otherwise) of any prefix used on the branch. We say a prefix ℓ' is a* ***simple extension*** *of ℓ if $\ell' = \ell,n$ for some integer n. (Fitting 1983:391.)*

Apart from these definitions we also require additional properties that allow us to determine the relationship between different prefixes. These properties need to mimic the behaviour of the relation R in the different standard models M=<W, R, V> for the normal systems of modal logic. In other words, if the relation R is transitive and reflexive then the prefixes need to mimic a transitive and reflexive relationship as well. In order to define this relationship between the prefixes we define an accessibility relation denoted by *accessible from*.

The accessibility relation, accessible from, for prefixed KT4 tableau systems have the following conditions (Fitting 1983: 393):

- Every prefix ℓ,n is accessible from ℓ for every integer n. For example 1,2 is accessible from 1.
- Every prefix ℓ is accessible from ℓ. For example 1 is accessible from 1.
- Every prefix ℓ,ℓ' is accessible from ℓ for every non-empty sequence ℓ'. For example 1,1,2 is accessible from 1,1 and 1.

The reader will recall that in chapter 3 we defined the class of KT5 models as the class of models M=<W, R, V> where R is the equivalence relation. This class can also be defined alternatively as the class of models M=<W, R, V> where R is the universal relation. It is this latter definition that forms the foundation of the *accessible from* relation for Fitting's (1983:397) prefixed KT5 tableau system. Essentially the idea is that since R is the universal relation every prefix is accessible from every other prefix. Thus for prefixed KT5 tableau systems the following condition has been defined for the *accessible from* relation:

- Any prefix ℓ is accessible from any other prefix ℓ'. For example, the prefix 2 is accessible from the prefix 3 and both prefixes 2 and 3 would be accessible from 1.

For the remainder of section 3, L will be used to denote the systems KT4 and KT5.

3.2 Rules

Similar to previous tableau systems, we also need to define specific tableau rules for prefixed tableau systems. Only the rules applicable to KT4 and KT5 will be addressed in this section. The reader is referred to Fitting (1983) for the rules for the other normal systems of modal logic. Once again we make use of all the previous propositional tableau expansion rules as defined in chapter 2 (see section 3.3) with the exception that we need to cater for an additional prefix. Note that the application of the α and β rules does not result in a change of worlds. For example, if A is a formula of type α that is true at a world named by ℓ then α_1 and α_2 are also true at the world named by ℓ. Denoted by:

$$\frac{\ell\alpha}{\begin{array}{c}\ell\alpha_1\\ \ell\alpha_2\end{array}}$$

We will not redefine the rules here, but care should be taken that when applying the tableau rules that the same prefix is carried forward.

For example, if the formula 1(P ∧ Q) occurs on the branch then the branch will be extended by the formulas 1P and 1Q.

The repetition rule, as defined previously, is also used in this system except that now the prefix is also copied with the formula.

The ν and π rules, defined below, are the same for both prefixed KT4 and KT5 tableau systems. However, the reader should note that although the rules look the same, the accessibility relation is different for these systems. It follows therefore that although the same rules are applicable to both tableau systems they will be *applied* differently.

- ν rule $\dfrac{\ell\nu}{\ell'\nu}$

  where ℓ' is accessible from ℓ and ℓ' has been used on the branch, or is an unrestricted simple extension of ℓ.

For example, if the formula 1□(P ∧ Q) occurs on the branch of a prefixed KT4 tableau then the branch can be extended by 1(P ∧ Q) since 1 has been used on the branch and 1 is *accessible from* 1. Alternatively, the branch can be extended by 1,2(P ∧ Q) provided that 1,2 is *accessible from* 1 and is an unrestricted simple extension of 1. From the conditions listed above for the *accessible from* relation we know that 1,2 is indeed accessible from 1. Also, it follows by definition 71, that 1,2 is a simple extension of 1. The only remaining condition that would have to be verified is that 1,2 has not been used as an initial segment (proper or otherwise) on the branch.

For a prefixed KT5 tableau system the application of the ν-rule to the formula 1□(P ∧ Q) would result in the branch being extended by the formula n(P ∧ Q) for any integer n that has either been used on the branch or is new to the branch. This follows from the condition of the *accessible from* relation that says that any prefix is accessible from any other prefix and from the ν rule that states that the prefix has either been used on the branch or is an unrestricted simple extension of another prefix. So in this case we could have extended the branch by: 1(P ∧ Q), 1,2(P ∧ Q) or just 2(P ∧ Q) etc.

- π rule $$\frac{\ell\pi}{\ell'\pi}$$

  where ℓ' is an unrestricted simple extension of ℓ.

For example, in a prefixed KT4 tableau system a tableau branch with the formula 3◇Q can be extended by the formula 3,4Q after the application of the π rule provided that the prefix 3,4 was an unrestricted simple extension of 3. If, for instance, the formula 3,4□P appeared somewhere on the branch then we would not have been able to use the prefix 3,4 but would have had to used another prefix such as 3,5, once again with the condition that 3,5 must be an unrestricted simple extension of 3.

In a prefixed KT5 tableau system the application of the π rule to the formula 3◇Q would result in the current branch being extended by the formula nQ for any *new* integer n. So in this case we could extend the branch by 4Q, 3,4Q etc. provided 4 and 3,4 have not appeared on the branch before. It should be clear to the reader that we would not have been able to extend the branch by 3Q since the prefix 3 has been previously used on the branch.

Global and local assumption rules, also defined previously, are also applicable to Fitting's (1983) explicit tableau system. In this case any member of the global assumptions can be added to a tableau branch as ℓ'A where ℓ' is a prefix that has been used on the branch. Members of the local assumptions are added as 1A to the tableau branch.

3.3 Soundness

Theorem *58 (Soundness)*

If A has a prefixed L-tableau proof, then A is valid in all L-models.

Proof: Proof is by contradiction. Assume that A has a L-tableau proof but that it is not valid in the classes of standard models corresponding to L. Show that there does not exist a labelled tableau for A, thus contradicting the initial assumption that A had a tableau proof. Details can be found in Fitting 1983:400,401. □

3.4 Completeness

3.4.1 *Systematic Tableau Procedure*

Although it is possible to establish the completeness of the various prefixed tableau systems by using maximal consistent sets, as we did with the previous tableau systems, we will use an alternative argument to prove the completeness for the prefixed tableau systems. This argument essentially makes use of a systematic-tableau construction procedure that describes how to build a tableau proof for a formula A, that is, we keep applying the rules in the systematic procedure until we either find a proof for A or a counter-model for A.

Fitting (1983) makes use of a device originated by Smullyan who worked with each occurrence of a prefixed formula only once, but whenever he worked with a formula of the form $\ell\nu$ he added a fresh occurrence of it at the end of the branch, that is, once a formula has been used it will be marked in some way to ensure that the prefixed formula is not used again except if the formula is of the type $\ell\nu$. A formula, which has been marked, will be called finished.

Only formulas that have the same prefix may be resolved with each other regardless of their markings. For example, if 1,2A and 1,3¬A appear on the same branch then the branch cannot be closed. However, if 1,2A and 1,2¬A marked finished, appear on the same branch then the branch will be closed.

Note that we will refer to a prefixed formula as atomic if it is of the form ℓA or ℓ¬A where A is any atomic formula.

Let A be a formula. We describe a systematic procedure to produce a proof for A in the logic L as follows. The proof is in stages (Fitting 1983:403, 404):

```
Stage 1      Begin by placing $1\neg A$ at the origin.
             This concludes stage 1.

             Suppose n stages of the construction have been completed.
             IF the tableau we have constructed is closed THEN
                Stop.
             END IF
             IF all occurrences of prefixed formula are finished THEN
                Stop.
             END IF
             Otherwise go on to stage $n + 1$.

Stage n+1    Choose an occurrence of a prefixed formula as high up in the tree as possible that has not
             been finished, for instance $\ell A$.
             IF $\ell A$ is atomic THEN
               Declare the occurrence finished.
               This ends stage n+1.
             END IF
             Otherwise extend the tableau as follows:
             FOR each open branch B through the occurrence of $\ell A$ DO:
                IF $\ell A$ is of the form $\ell\alpha$ THEN
                   Add $\ell\alpha_1$ and $\ell\alpha_2$ to the end of B.
                END IF
                IF $\ell A$ is of the form $\ell\beta$ THEN
                   Split the end of branch B into two.
                   Add $\ell\beta_1$ to the end of one branch
                   Add $\ell\beta_2$ to the end of the second branch.
                END IF
                IF $\ell A$ is of the form $\ell\nu$ THEN
                   FOR each prefix $\ell'$ that has been used on B and that is accessible from $\ell$ (if any) DO
                      Add $\ell'\nu_0$ to the end of B
                   END FOR
                   Add a fresh occurrence of $\ell\nu$ to the end of B.
                END IF


                IF $\ell A$ is of the form $\ell\pi$ THEN
```

```
        Let k be the smallest integer such that $\ell$,k is unrestricted on B
        Add $\ell$,k$\pi_0$ to the end of B.
    END IF
END FOR
```

Having done this for each branch B through the particular occurrence of ℓA being considered, declare that occurrence of ℓA finished.

This completes state n+1.

3.4.2 *Completeness Proofs*

Theorem 59

If a formula A is L-valid, then A has a proof in the prefixed L-tableau system.

Proof: The proof follows a contrapositive argument, that is, we assume that A is not provable and then show that a systematic attempt to prove A, using the procedure above, will fail. From this result it follows that there is a tableau branch that is still open and as a result all the formulas on the open branch (including ¬A) are satisfied by the class of models corresponding to L, that is, there is some model M=<W, R, V> in the class of models corresponding to L and a world $\alpha \in W$ such that all the formulas on the branch, including ¬A, are true at α. This in turn means that A is not L-valid. We refer the reader to Fitting (1983:410) for the details.□

Theorem 60

If A has an L-tableau proof, then A has a systematic L-tableau proof.

Proof: Follows as a corollary of the above theorem (Fitting 1983:410).□

From the above two theorems we can therefore conclude that if A is L-valid then A has a proof in a prefixed L-tableau system which in turn means A has a systematic L-tableau proof.

3.5 Applications.

The systematic procedure is demonstrated below for KT4 and KT5.

For KT4 we show that the formula $\Box P \to \Box\Box P$ is valid in the class of standard models corresponding to KT4. The formula is negated and the tableau is created as follows.

Note that the tableau is in the centre; the other columns are there for clarification. The leftmost column indicates the stage at which the formula was created. For instance, $1\Box P$ was created during stage 2. The next column indicates which rule was applied to create the new formula. For example, $1\Box P$ was created by the α rule that was applied to the formula $1\neg(\Box P \to \Box\Box P)$ during stage 2. The right most column indicates at which stage the formula was marked finished. For example, $1\neg(\Box P \to \Box\Box P)$ was marked finished at stage 2.

1		$1\neg(\Box P \to \Box\Box P)$	finished	2
2	α rule $(1\neg(\Box P \to \Box\Box P))$	$1\Box P$	finished	3
2	α rule $(1\neg(\Box P \to \Box\Box P))$	$1\neg\Box\Box P$	finished	4
3	ν rule $(1\Box P)$	$1P$	finished	5
3		$1\Box P$	finished	6
4	π rule $(1\neg\Box\Box P)$	$1,2\neg\Box P$	finished	7
6	ν rule $(1\Box P)$	$1P$	finished	8
6	ν rule $(1\Box P)$	$1,2$	finished	9
6		$1\Box P$	finished	10
7	π rule $(1,2\neg\Box P)$	$1,2,3\neg P$		
10	ν rule $(1\Box P)$	$1P$		
10	ν rule $(1\Box P)$	$1,2$		
10	ν rule $(1\Box P)$	$1,2,3P$		
10		$1\Box P$		

Clearly this tableau is closed because both $1,2,3\neg P$ and $1,2,3P$ are both on the only branch in the tableau.

For KT5 the formula $\Diamond P \rightarrow \Box\Diamond P$ is shown to be valid in the class of standard models corresponding to KT5 (Fitting 1983:397). Once again the proof follows a similar outline as the one described above.

1		$1\neg(\Diamond P \rightarrow \Box\Diamond P$	finished	2
2	α rule($1\neg(\Diamond P \rightarrow \Box\Diamond P$	$1\Diamond P$	finished	3
2	α rule ($1\neg(\Diamond P \rightarrow \Box\Diamond P)$	$1\neg\Box\Diamond P$	finished	4
3	π rule ($1\Diamond P$	2	finished	5
4	π rule ($\neg\Box\Diamond P$	$3\neg\Diamond P$	finished	6
6	v rule ($\neg\Diamond P$	$1\neg P$		
6	v rule ($\neg\Diamond P$	$2\neg P$		
6	v rule ($\neg\Diamond P$	$3\neg P$		

This tableau is closed since $2\neg P$ and 2P are both on the only branch in the tableau.

3.6 Decidability

At this stage there is no guarantee that the systematic procedure, as given above, will ever terminate. In other words it is very likely that if there is not a proof for A that this procedure may carry on forever. As a result the tableau that is under construction will be infinite. This in turn means that since we have an infinite tableau, with a finite number of branches, at least one of tableau's branches must be infinite (see Fitting 1983: 406 for proof).

It follows therefore from the above discussion that the systematic tableau procedure needs to be modified in order to ensure that the procedure does in fact terminate with either a counter-model or a proof for a given formula A. These modifications are described as follows:

- In the FOR loop for stage n + 1, add the following note to the end of all statements that start with the word ADD: *provided an occurrence of it is not already present on the branch.* For example the ADD statement for the ν rule will become: Add $\ell'\nu_0$ to the end of B provided an occurrence of it is not already present on the branch.

- We alter the existing ν rule with the following:

```
IF $\ell A$ is of the form $\ell\nu$ THEN
    FOR each prefix $\ell'$ that has been used on B and that is accessible from $\ell$ (if any) DO
        Add $\ell'\nu_0$ to the end of B provided an occurrence of it is not already present on the branch
    END FOR.
END IF.
```

In other words, we do not continually add an occurrence of $\ell\nu$ to the end of the branch.

Fitting (1983:411-412) shows that these changes are sufficient to show decidability for logics that do not involve transitivity. Once logics involving transitivity, such as KT4 and KT5, are considered the systematic procedure given above may not terminate and therefore further alterations to the procedure are required. Before these changes can be addressed the following technical information is required (Fitting 1983:413, 414).

Definition 72

By a ***chain*** *of prefixes we mean a sequence* $1, \ell_1, \ell_2, \ldots$ *in which each is a simple extension of the preceding.*

Definition 73

A chain of prefixes $1, \ell_1, \ell_2, \ldots$ *from a branch B is* ***periodic*** *if there exists distinct prefixes* ℓ_i *and* ℓ_j *in the chain* $(i < j)$ *such that* $\ell_i A$ *(A is any formula) is on B iff* $\ell_j A$ *is on B, that is, if* $\{A \mid \ell_i A \text{ on } B\} = \{A' \mid \ell_j A' \text{ on } B\}$*. A branch is periodic if every infinite chain (of prefixes) on B is periodic.*

Definition 74

For a particular prefix ℓ *and a branch B of a tableau system, we say that the set of formulas* ***B-associated*** *with* ℓ *is the set of all formulas A such that* ℓA *occurs on B.*

Suppose that we are trying to construct a systematic tableau for a formula A, that is, we start constructing a systematic tableau with 1A as the root node. It follows therefore, that if any prefixed formula ℓZ occurs on a branch B, that Z must be some subformula of A or the negation of some subformula of A. So let $\Gamma(A)$ be the collection of all the subformulas of a formula A and all the negations of these subformulas. $\Gamma(A)$ is finite, say of size n, since there are only a finite number of subformulas for any formula A and only a finite number of negations of these subformulas. It follows therefore that if a subformula of A, say Z, occurs on B as ℓZ then Z or $\neg Z$ must belong to $\Gamma(A)$. In other words, the set B-associated with a given prefix ℓ must be a subset of $\Gamma(A)$. Thus there are at most 2^n possibilities for the set of formulas B-associated with the various prefixes on B. It follows that, for any chain of prefixes $1, \ell_1, \ell_2, \ldots$ occurring on B, there must be two, ℓ_i and ℓ_j where $i < j \leq 2^n$, such that both have the same set of formulas B-associated with them, that is, the set of formulas B-associated with ℓ_i, $\{Z \mid \ell_i Z \text{ is on } B\}$, equals the set of formulas B-associated with ℓ_j, $\{Z \mid \ell_j Z \text{ is on } B\}$. The same result holds true for the sets B-associated with the prefixes ℓ_{i+k} and ℓ_{j+k} for any integer $k \geq 1$. This in turn, by definition 73, means that the chain of prefixes after the prefix ℓ_j becomes periodic.

Suppose for each chain $1, \ell_1, \ell_2, \ldots$ that occurs on B (where 1 is the prefix of the root node), we only keep the initial segment up to where it begins to become periodic and we discard the remainder of the segment, that is, if ℓ_i and ℓ_j $(i < j)$ have the same sets of B-associated formulas, and i and j are the smallest numbers for which this is true, we retain only $1, \ell_1, \ell_2, \ldots, \ell_i, \ldots, \ell_j$ and discard the rest. Then it follows that the chain on B will be finite and of maximum length 2^n. If we repeat this exercise for each chain on B, then every chain on B will be finite and of maximum length 2^n. If all the chains on B are finite B will also be finite.

Note that the finite branch B also guarantees us the existence of a counter-model, that is, if our construction has become periodic, and we have not had a closure, we will never get it by continuing with B. If we do continue with B we will construct an infinite branch and hence a counter-model.

The trick now is to determine when a branch B becomes periodic. Fortunately we can do this by looking at the prefixes of the formulas. For any given prefix k, we can tell when all the formulas prefixed by k have been introduced to the branch B. This is trivial if k is of length 1. Formulas with prefixes of length k+1 can only be introduced onto B, by using the π-rule on formulas with prefixes of length k that are already on B. Once all such formulas have been declared finished, the only new formulas with prefixes of length k+1 that can be added to B, will be by means of the ν, α, β rules. Clearly we will know when all the appropriate applications of these rules will have been made. It follows therefore that for any branch we will know when all the possible prefixed formulas, of a given prefix length, are present, that is, we will be able to tell that a branch has become periodic, without constructing a branch infinitely. (Fitting 1983:413-416.)

In lieu of the above results, we can now return to the systematic proof procedure and make the following alternations:

Declare all prefixed formulas of the form ℓA on a branch finished if:

1. we know these are all the formulas prefixed by ℓ that we can get or

2. if the set of formulas associated with ℓ on the branch duplicates the set associated with some proper initial segment of ℓ.

With this modification the systematic tableau procedure must terminate, either in a closed tableau or with a counter-model.

4. TABLEAU SYSTEMS - RAJEEV GORé

In this section we will provide a brief overview of the semantic tableaux methods as described by Goré (1995). Goré's work has been based largely on that of Hintikka (1955) and Rautenberg (1983) with reference to a number of other researchers in particular Fitting (1983). We will focus on Goré's application as it relates to the implicit and explicit tableau systems (as defined below) of KT4, KT4L and KT5.

Implicit tableau systems are dealt with first. In implicit systems the properties of the reachability relation R, such as reflexivity and transitivity, are built into the tableaux rules, that is, we do not reason explicitly about R. These systems are similar to Fittings systems discussed in Section 2.

4.1 Syntax and Semantics

Gor$_é$'s (1995) tableau system only uses finite sets of formulas which are defined in terms of the adequate set of connectives □, ¬ and ∧. We therefore assume that the formulas are in this format for the remainder of this section.

Note that when it is convenient to do so, we will use L instead of KT4, KT4L or KT5, that is, L should be read as KT4, KT4L or KT5. We define the tableau rules in terms of Gor$_é$'s (1995) notation as follows (Gor$_é$ 1995:12):

Definition 75

A tableau rule τ consists of a numerator N (above the line) and a (finite) list of denominators D_1, D_2, ..., D_k (below the line) separated by vertical bars. Denoted:

$$(\tau)\frac{N}{D_1|D_2|\ldots|D_k}$$

The numerator N, of each tableau rule contains one or more distinguishable formulas called the *principal* formulas. Each denominator D_i, usually contains one or more distinguishable formulas called the *side* formulas. The (τ) next to the tableau rule denotes the name of the tableau rule and is usually associated with the main connective but may have a more complex name. For example, consider the (¬) rule below: This rule has a numerator Γ;¬¬A with principal formula ¬¬A and one denominator Γ;A with A as the side formula:

$$(\neg)\frac{\Gamma;\neg\neg A}{\Gamma;A}$$

Note that Γ is a finite set of formulas, that is, effectively we could rewrite the numerator as Γ ∪ {¬¬A} and the denominator as Γ ∪ {A} where Γ is any set of formulas say $\{A_1, A_2, ..., A_n\}$. So if we had the set of formulas {P, Q, ¬¬R} we could apply the rule (¬) and replace this set with the set {P, Q, R}. In this example Γ would be the set {P, Q}.

Definition 76

*A **tableau system** (or calculus) CL is a finite collection of tableau rules τ_1, τ_2, ..., τ_m identified with the set of its rule names; thus CL = {τ_1, τ_2, ..., τ_m }, where L is either KT4, KT4L or KT5. A **CL-tableau** for Γ is a finite tree with root Γ whose nodes carry finite formula sets (Gor$_é$ 1995:13).*

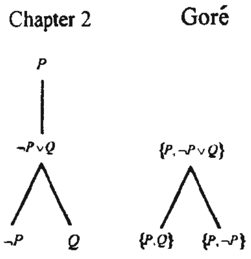
We extend a tableau using Goré's (1995) method as follows:

1. Choose a leaf node n with a set of formulas Γ, where n is not an end node (defined below), and choose a tableau rule τ such that $\Gamma = N$, the numerator of τ.

2. If τ has k denominators then create k successor nodes for n, with successor i carrying an appropriate instantiation of a denominator D_i.

3. If after executing steps 1 and 2 a successor node s, carries a set Γ' such that Γ' has already appeared on the branch from the root to s, then s is an end node.

It should be clear from the definition that there is a set of formulas per node rather than just one formula as discussed in chapter 2 (section 3.3). For instance, instead of having P, ¬P∨ Q, as two nodes on a branch, we will group them as a set {P, ¬P ∨ Q} into one node. As we apply the tableau rules we will alter the set according to the rules and carry the set forward to the additional nodes. For example, in chapter 2 (section 3) if we had applied the β rule to ¬P ∨ Q we would have created two new nodes one for ¬P and one for Q. In Goré's (1995) tableau we will also create two nodes, the first node will contain the set of formulas {P, ¬P} and the second node the set of formulas {P, Q}. We illustrate this as follows:



We define closure, in terms of Goré (1995) as follows:

Definition 77

A branch B in a tableau is ***closed*** *if its end node is {⊥}; otherwise it is* ***open****. A tableau is closed if all its branches are closed; otherwise it is open (Goré 1995:13).*

Definition 78

A formula A is a ***theorem*** *of CL if there is a closed tableau for the set {¬A}, denoted $\square_{CL}A$ (Goré 1995:13).*

Goré's tableau system ensures that the notion of theoremhood can be extended to the notion of deducibility, denoted by $\Gamma \Box_{CL} A$, that is, the *deduction theorem*: $\Gamma \Box_{CL}(A \rightarrow B)$ if and only if $\Gamma \cup \{A\} \Box_{L} B$ is preserved by his system. (Goré 1995:4.)

Note that in this system if there is a closed tableau for the set $\Gamma \cup \{\neg A\}$ then we can conclude $\Gamma \Box_{C} A$ where C is the class of standard models corresponding to some normal system L, A is any formula, Γ is any finite set of formulas, that may be empty and $\Box$ is local logical entailment.

4.2 Rules

In CL we distinguish between static and transition rules. By static rules we mean rules that do not change the worlds at which formulas are true. With transitional rules we change the scope or world for a given formula. Intuitively the transition rules are based on the definition of the □. So for instance, if we have □A in a L-model = <W,R,V> at $\alpha \in W$ we know that for every world $\beta \in W$ such that $\alpha R \beta$, A is true at β. So if we have a tableau rule that performs this shift from α to β we have not changed the satisfiability of a formula A, but merely shifted its context. The following static and transitional rules are available to a CKT4, CKT4L and CKT5.

4.2.1 *Tableau Rules for Propositional Logic*

We define the rules available to all propositional tableau systems (denoted CPC) in terms of Goré's (1995) notation:

$$(\theta)\frac{\Gamma;\Gamma'}{\Gamma} \qquad (\bot)\frac{\Gamma;A;\neg A}{\bot}$$

$$(\neg)\frac{\Gamma;\neg\neg A}{\Gamma;A} \qquad (\wedge)\frac{\Gamma;A\wedge B}{\Gamma;A;B}$$

$$(\vee)\frac{\Gamma;\neg(A\wedge B)}{\Gamma;\neg A|\Gamma;\neg B}$$

where $\Gamma \neq \Gamma'$ are any sets of formulas and A and B are any formulas. As discussed previously in order to apply these rules we must match the numerator of these rules to some node and then create a new node as specified by the denominator. For instance, if we are at a node that contains the set $\{\Box P, \Diamond Q, R, \neg R\}$ we can apply the ⊥ rule to get a new node containing ⊥. In this example the numerator is the union of the sets $\Gamma = \{\Box P, \Diamond Q\}$, $A = \{R\}$ and $\neg A = \{\neg R\}$.

4.2.2 Tableau Rules for CKT4

Static Rules: CPC

$$(T)\frac{\Gamma;\Box A}{\Gamma;\Box A;A}$$

Where Γ is a finite set of formulas that may be empty and A is any formula.

Applying the T rule to the set $\{\Box P, \Box(P \wedge Q), \neg P\}$ where $\Gamma = \{\Box P, \neg P\}$ and $\Box A = \Box P$ results in the following set $\{\Box P, \Box(P \wedge Q), \neg P, P\}$ where $\Gamma = \{\Box P, \neg P\}$, $\Box A = \Box P$ and $A = P$.

Transitional Rules:

$$(KT4)\frac{\Box\Gamma;\neg\Box A}{\Box\Gamma;\neg A}$$

Where Γ is any set of formulas, A any formula and if Γ denotes the set $\{A_1, A_2, \ldots, A_n\}$ then $\Box\Gamma$ denotes the set $\{\Box A_1, \Box A_2, \ldots, \Box A_n\}$, that is, the set $\Box\Gamma$ denotes all the formulas that occur in the numerator that are of the form $\Box A$. For example, if we have the set of formulas $\{\Box P, \Box(P\wedge Q), \neg P\}$ then $\Box\Gamma = \{\Box P, \Box(P \wedge Q)\}$.

For example, applying the KT4 rule to the set $\{\Box P, \neg\Box R\}$ results in the set $\{\Box P, \neg R\}$ where $\Box\Gamma = \{\Box P\}$, $\neg\Box A = \neg\Box R$ and $\neg A = \neg R$. Also note that since we have to match the numerator with the set of formulas at the current node, we may have to apply the *(θ)* rule to reduce the initial set to a set that matches the numerator $\Box\Gamma \cup \neg\Box A$. For example, if the set at the current node is the set $\{\Box P, \neg\Box R, Q\}$ then applying the *(θ)* rule to this set will result in the set $\{\Box P, \neg\Box R\}$ to which we can now apply the KT4 rule.

4.2.3 *Tableau Rules for CKT4L*

Static Rules: CPC

(T)

Transitional Rules:

$$(KT4L)\frac{\Box\Gamma;\{\neg\Box A_1,\neg\Box A_2,\ldots,\neg\Box A_k\}}{\Box\Gamma;\overline{\Gamma_1'};\neg A_1\,\big|\ldots\big|\,\Box\Gamma;\overline{\Gamma_k'};\neg A_k}$$

where $\Gamma' = \{\neg\Box A_1, \neg\Box A_2, \ldots, \neg\Box A_k\}$ and $\overline{\Gamma'_i} = \Gamma'$ without $\{\neg\Box A_i\}$ and $\Box\Gamma$ is a set of formulas as defined above.

For example, applying the (KT4L) rule to the set $\{\Box(P \wedge Q), \neg\Box P, \neg\Box Q\}$, where $\Box\Gamma = \{\Box(P \wedge Q)\}$ and $\{\neg\Box A_1, \neg\Box A_2\} = \{\neg\Box P, \neg\Box Q\}$, results in the sets $\{\Box(P \wedge Q), \neg\Box Q, \neg P\}$and $\{\Box(P \wedge Q), \neg\Box P, \neg Q\}$ where $\Box\Gamma = \{\Box(P \wedge Q)\}$, $\overline{\Gamma'_1} = \{\neg\Box Q\}$, $\neg A_1 = \{\neg P\}$, $\overline{\Gamma'_2} = \{\neg\Box P\}$ and $\neg A_2 = \{\neg Q\}$.

4.2.4 *Tableau Rules for CKT5π*

Although Goré (1995) considers a number of different CKT5 systems for KT5 we will only look at his CKT5π system. This system is similar to Fittings (1983) semi-analytic tableau system for KT5.

Static Rules: CPC

(T)

Transitional Rules: $$(KT5)\frac{\square\Gamma; \neg\square\Gamma'; \neg\square A}{\square\Gamma; \neg\square\Gamma'; \neg\square A; \neg A}$$

Where A is any formula □Γ is as defined above and ¬□Γ' = {¬□A_1, ¬□A_2, ... ¬□A_n}, where Γ' = {A_1, A_2, ..., A_n}.

For example, if the (KT5) rule is applied to the set {□P, □(P ∧ Q), ¬□Q, ¬□¬P, ¬□(P ∨ Q)} where □Γ = {□P, □(P ∧ Q)}, ¬□Γ' = {¬□Q, ¬□¬P} and ¬□A = {¬□(P ∨ Q)} then the resulting set will be {□P, □(P ∧ Q), ¬□Q, ¬□¬P, ¬□(P ∨ Q), ¬(P ∨ Q)} where □Γ, ¬□Γ', ¬□A are as defined previously and ¬A = {¬(P ∨ Q)}. Note that this rule is not deterministic, that is, there is more than one way to apply this rule given the initial set. For instance, we could have chosen ¬□A = {¬□Q} or ¬□A = {¬□¬P}.

4.3 Soundness

Theorem *61 (CL Soundness)*

Each calculus CL is sound with respect to the class of standard models corresponding to L, that is, for any formula A and any finite set of formulas Γ, $\Gamma\square_{CL}A$ implies $\Gamma\square_{CL} A$.

Proof: Essentially the proof involves showing for each rule in CL that if the numerator of the rule is satisfiable in the class of standard models corresponding to L then so is at least one of the denominators. Clearly the CPC rules are sound since each world behaves classically. The modal rules can be shown to be sound with respect to some known property of R as enforced by the class of standard models corresponding to L. (Goré 1995:28-31, 50.)□

4.4 Completeness

Theorem *62 (CL Completeness)*

Each calculus CL is complete with respect to the class of standard models corresponding to L, that is, for any formula A and any finite set of formulas Γ, $\Gamma\square_{CL}A$ implies $\Gamma\square_{CL}A$.

Proof: The proof is by contrapositive argument, that is, assume that it is not the case that $\Gamma\square_{CL}A$ and show that is not the case that $\Gamma\ \square_{CL}A$. Since it is not the case that $\Gamma\square_{CL}A$ no tableau for Γ ∪ {¬A} closes.

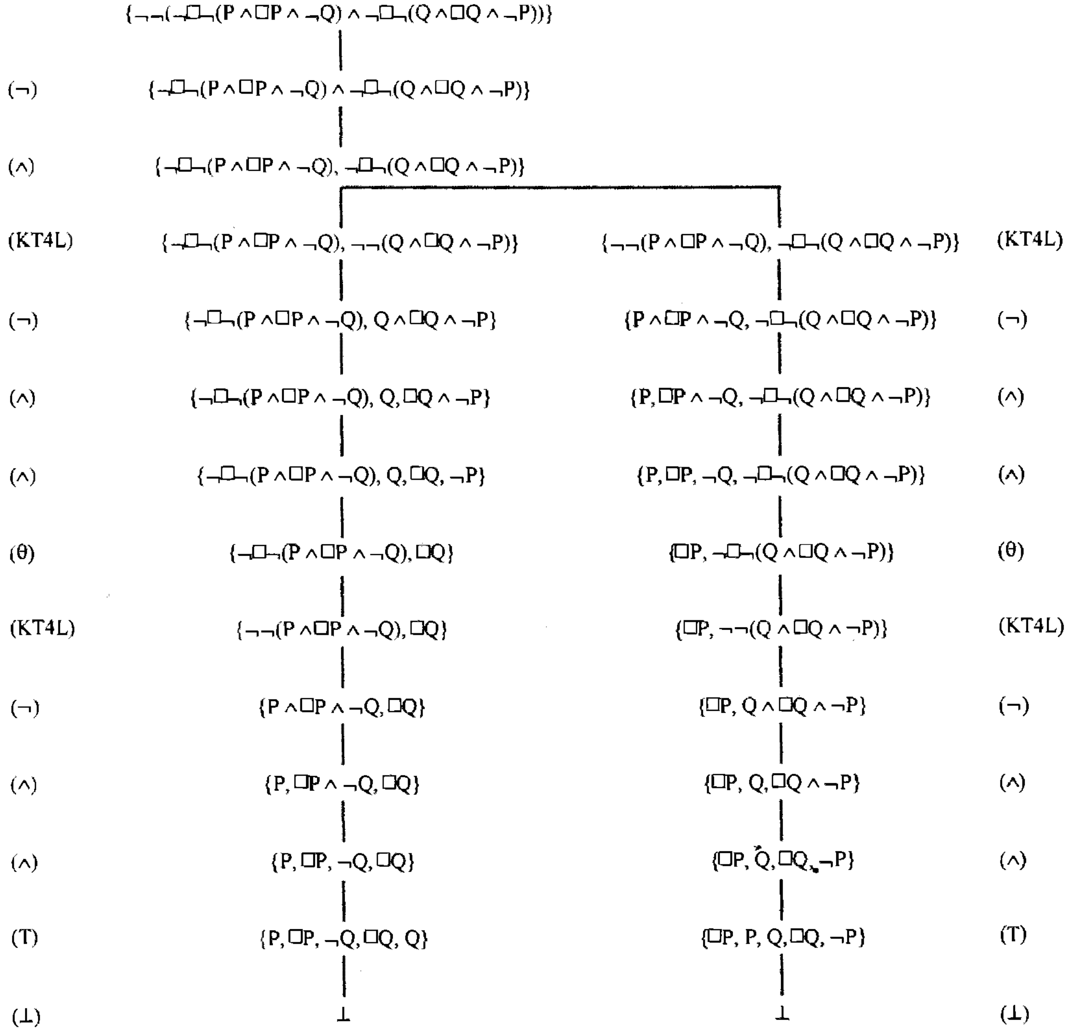
Essentially we pick and choose sets with certain special properties from possible different open tableaux for $\Gamma \cup \{\neg A\}$ and use them as possible worlds to construct an L-model M for $\Gamma \cup \{\neg A\}$, safe in the knowledge that each of these sets is satisfiable. The model M is constructed in such a way that it contains a world α such that $\square_{\alpha}^{M} \Gamma$ and $\square_{\alpha}^{M} \neg A$. Hence we demonstrate by construction that it is not the case that $\Gamma \square_{LC} A$. See Goré (1995:36,43,52,53) for details. □

4.5 Applications

In this example the rules for the KT4 tableau system are demonstrated by showing that the formula $\neg(\square P \wedge \neg\square\square P)$ is a theorem of KT4. Negating the formula the proof follows. Note that justifications for the sets at the different nodes are given on the left of the tableau.

$\{\neg\neg(\square P \wedge \neg\square\square P)\}$	
|	
$\{\square P \wedge \neg\square\square P$	by the $\neg$ rule
|	
$\{\square P, \neg\square\square P$	by the $\wedge$ rule
|	
$\{\square P, \neg\square P$	by the KT4 rule
|	
$\{\square P, \neg P$	by the KT4 rule
|	
$\{\square P, P, \neg P$	by the T rule
|	
$\perp$	by the $\perp$ rule

In this example the rules for the system CKT4L are demonstrated by showing that the formula (¬(¬□¬(P ∧ □P ∧ ¬Q) ∧ ¬□¬(Q ∧ □Q ∧ ¬P))) is a theorem of KT4L.


{¬¬(¬□¬(P ∧□P ∧ ¬Q) ∧ ¬□¬(Q ∧□Q ∧ ¬P))}
(¬) {¬□¬(P ∧□P ∧ ¬Q) ∧ ¬□¬(Q ∧□Q ∧ ¬P)}
(∧) {¬□¬(P ∧□P ∧ ¬Q), ¬□¬(Q ∧□Q ∧ ¬P)}
(KT4L) {¬□¬(P ∧□P ∧ ¬Q), ¬¬(Q ∧□Q ∧ ¬P)}
{¬¬(P ∧□P ∧ ¬Q), ¬□¬(Q ∧□Q ∧ ¬P)} (KT4L)
(¬) {¬□¬(P ∧□P ∧ ¬Q), Q ∧□Q ∧ ¬P}
{P ∧□P ∧ ¬Q, ¬□¬(Q ∧□Q ∧ ¬P)} (¬)
(∧) {¬□¬(P ∧□P ∧ ¬Q), Q, □Q ∧ ¬P}
{P, □P ∧ ¬Q, ¬□¬(Q ∧□Q ∧ ¬P)} (∧)
(∧) {¬□¬(P ∧□P ∧ ¬Q), Q, □Q, ¬P}
{P, □P, ¬Q, ¬□¬(Q ∧□Q ∧ ¬P)} (∧)
(θ) {¬□¬(P ∧□P ∧ ¬Q), □Q}
{□P, ¬□¬(Q ∧□Q ∧ ¬P)} (θ)
(KT4L) {¬¬(P ∧□P ∧ ¬Q), □Q}
{□P, ¬¬(Q ∧□Q ∧ ¬P)} (KT4L)
(¬) {P ∧□P ∧ ¬Q, □Q}
{□P, Q ∧□Q ∧ ¬P} (¬)
(∧) {P, □P ∧ ¬Q, □Q}
{□P, Q, □Q ∧ ¬P} (∧)
(∧) {P, □P, ¬Q, □Q}
{□P, Q, □Q, ¬P} (∧)
(T) {P, □P, ¬Q, □Q, Q}
{□P, P, Q, □Q, ¬P} (T)
(⊥) ⊥
⊥ (⊥)

In the final example the rules for CKT5 are demonstrated by showing that the formula $\neg(\neg\Box\neg P \wedge \neg\Box\neg\Box\neg P)$ is a theorem of KT5.

$$\begin{array}{cl}
\{\neg\neg(\neg\Box\neg P \wedge \neg\Box\neg\Box\neg P)\} & \\
| & \\
\{\neg\Box\neg P \wedge \neg\Box\neg\Box\neg P\} & \text{by the } \neg \text{ rule} \\
| & \\
\{\neg\Box\neg P, \neg\Box\neg\Box\neg P\} & \text{by the } \wedge \text{ rule} \\
| & \\
\{\neg\Box\neg P, \neg\Box\neg\Box\neg P, \neg\neg\Box\neg P\} & \text{by the KT5 rule} \\
| & \\
\{\neg\Box\neg P, \neg\Box\neg\Box\neg P, \Box\neg P\} & \text{by the } \neg \text{ rule} \\
| & \\
\{\Box\neg P, \neg\Box\neg\Box\neg P, \neg\Box\neg P, \neg\neg P\} & \text{by the KT5 rule} \\
| & \\
\{\Box\neg P, \neg\Box\neg\Box\neg P, \neg\Box\neg P, P\} & \text{by the } \neg \text{ rule} \\
| & \\
\{\neg\Box\neg\Box\neg P, \neg\Box\neg P, P, \Box\neg P, \neg P\} & \text{by the T rule} \\
| & \\
\bot & \text{by the } \bot \text{ rule}
\end{array}$$

4.6 Decidability

We require the following results in order to show decidability.

Definition 79

A finite set of formulas Γ, and a formula A is ***CL-inconsistent*** *if the CL-tableau for $\Gamma \cup \{\neg A\}$ is closed, denoted $\Gamma\Box_{CL}A$, otherwise it is* ***CL-consistent*** *(Goré 1995:13).*

Definition 80

A set Γ is closed with respect to a tableau rule if, whenever the numerator of the rule is in Γ so is at least one of the denominators of the rule. A set Γ is ***CL-saturated*** *if it is CL-consistent and closed with respect to the static rules of CL. (Goré 1995:22.)*

Theorem *63*

If there is a closed CL-tableau for the finite set of Γ then there is a closed CL-tableau for Γ with all the nodes in some finite set Γ^*_{CL} (where the nodes in Γ^*_{CL} are either from Γ or as the result of one of the rules of CL).

(See Gorė 1995:20 for proof.)

The reader will recall that in chapter 2 we defined a frame as a model M without the mapping V, that is, F = <W, R> is a frame. For the sake of convenience we will generalise the term frame to *L-frame* where L is KT4, KT4L or KT5. Similarly we will refer to an *L-model* instead of a KT4 model, KT4L model or KT5 model.

Definition 81

*A **model graph** for some finite fixed set of formulas Γ is a finite L-frame <W,R> such that all $\alpha \in W$ are CL-saturated sets with $\alpha \subseteq \Gamma^*_{CL}$ and*

- *$\Gamma \subseteq \alpha$ for some $\alpha \in W$*
- *if $\neg\Box A \in \alpha$ then there exists some $\beta \in W$ such that $\alpha R \beta$ and $\neg A \in \beta$.*
- *If $\alpha R \beta$ and $\Box A \in \alpha$ then $A \in \beta$. (Gorė 1995:23.)*

Theorem *64*

If <W,R> is a model graph for Γ then there exists an L-Model for Γ.

(See Gorė 1995:23 for proof.)

Recall from chapter 3 that we showed that if a modal logic say L is axiomatisable and has the finite model property then L is decidable. From the above results we have that CL is axiomatisable and therefore all we need to show is that CL contains the finite model property. We can do this by constructing a model graph for some finite fixed set of formula Γ If the L-model (defined in the above theorem) is chosen as finite for a finite Γ then L has the finite model property. Hence it follows that CL provides a decision procedure for $\Gamma \Box_L A$, for any formula A.

5. EXPLICIT TABLEAU SYSTEMS - RAJEEV GOR$_{\acute{e}}$

5.1 Syntax and Semantics

In the ensuing sections we will focus on the syntax and semantics of explicit tableau systems for KT4 and KT5. L should therefore be read as KT4 or KT5. As with Fitting's (1983) explicit tableau system the relation R is represented explicitly by some device.

Gor$_{\acute{e}}$ (1995) distinguishes between two different ways of representing a reachability relation R. One way is to maintain a network with named nodes, where each node contains a set of formulas. In this scenario a separate relation R(x, y) is also kept to represent the fact that the node named y is reachable from the node named x, where x and y are indices to allow cross-referencing between these two 'data structures'.

The second option, which is the option we will follow in this section, is to incorporate complex or structured world names into the syntax. Essentially the approach uses a global set of integers to label different formulas at different worlds, in order to distinguish these formulas from each other and their different worlds. For example, if a formula is true at a world ℓ then we label the formula with ℓ. The reachability relation is therefore built into the structure of the labels, avoiding the necessity of keeping a separate reachability data structure. Note that apart from a couple of alterations here and there Gor$_{\acute{e}}$'s explicit tableau system is similar to Fitting's (1983) prefixed tableau system discussed in section 3.

Definition 82

*A **label** (denoted ℓ or ℓ_i) is a non empty sequence of positive integers separated by dots. The **length** of a label ℓ is the number of integers it contains. A label ℓ is a **simple extension** of a label ℓ' if $\ell = \ell'.n$ for some $n \geq 1$. A label ℓ is an **extension** of label ℓ' if $\ell = \ell'.n_1.n_2. \ldots .n_k$ for some $k \geq 1$ with each $n_i \geq 1$. (Gor$_{\acute{e}}$ 1995:74.)*

For example 1, 1.2 and 1.2.3 are three labels with lengths 1, 2 and 3 respectively. We will use $\boldsymbol{L}$ to denote a set of labels and may omit the dots for convenience. For instance, instead of $\ell_1.\ell_2$ we will just write $\ell_1\ell_2$.

Definition 83

A set of labels L *is* ***strongly generated*** *(with root* ℓ_r*) if:*

1. *There is some root labelled* $\ell_r \in L$ *such that every other label in* L *is an extension of* ℓ_r*; and*
2. $\ell_i.n \in L$ *implies* $\ell_i \in L$ *(Goré 1995:74.)*

Intuitively the labels capture a basic reachability relation between the worlds they name. For instance the world labelled $\ell.1$ is accessible from the world named by ℓ. Strongly generated labels can also be viewed as trees with root node ℓ_r where a node labelled $\ell.1$ is the immediate child of a node labelled ℓ.

Definition 84

A ***labelled formula*** *is a structure of the form* $\ell::A$ *where* ℓ *is a label and* A *is a formula. A* ***labelled tableau rule*** *has a numerator and one or more denominators as before except that each numerator is comprised of a single labelled formula, and each denominator is comprised of at most two labelled formulas. A* ***labelled tableau calculus*** *(denoted LC) is simply a collection of labelled tableau rules. (Goré 1995:74, 75.)*

We redefine closure in terms of LC.

Definition 85

A ***labelled tableau*** *for a finite set of formulas* $\Gamma=\{A_1, A_2, ..., A_n\}$ *is a tree, where each node contains a single labelled formula. A* ***tableau branch*** *is any path from the root downwards in such a tree. A* ***branch is closed*** *if it contains some labelled formula* $\ell::A$ *and also contains* $\ell::\neg A$*. Otherwise it is open. A* ***tableau is closed*** *if every branch is closed, otherwise it is* ***open****. (Goré 1995:77.)*

It follows therefore that if we find a closed tableau for a $\Gamma \cup \{\neg A\}$ then we can conclude $\Gamma \square_C A$ where Γ is a finite set of formulas possibly empty, A is any formula, C is the class of standard models corresponding to L and $\square$ is local logical entailment.

Definition 86

A label ℓ *is* ***used*** *on a branch if there is some labelled formula* $\ell::A$ *on that branch. A label* ℓ *is* ***new*** *to a branch if there is no labelled formula* $\ell::A$ *on that branch (Goré 1995:77).*

If Γ_L is a set of labelled formulas then we let lab(Γ_L) = $\{\ell | \ell::A \in \Gamma_L\}$ be the set of labels that appear in Γ_L. If we refer to a branch B as the set of labelled formulas on the branch, then lab(B) is just the set of labels that is used on branch B.

5.2 Rules

As with implicit tableau systems we can define a number of different rules for explicit tableau systems: We will distinguish between three types (Goré 1995).

1. LCPC rules: The rules that are needed by any propositional logic (denoted PC). Labels in the denominator and numerator of these rules are identical.

2. ν-rules: All the rules applicable to formulas of the form ℓ::□A. These rules add formulas in the denominator to the already existing worlds named by the label of the denominator.

3. π-rules: All the rules applicable to formulas of the form ℓ::¬□A. These rules are known as 'successor creators' since they are the only rules that are permitted to create new worlds.

In the tableau rules below note that the world named by the label in the denominator is at most one step away from that named by the numerator. Also note, that apart form the (π-rule), all labels in the denominator must already exist on the branch, that is, the (π-rule) is the only rule that can add new labels.

5.2.1 Tableau Rules for LCPC

The following rules are available to all LC propositional logic systems.

$$(l\neg)\frac{\ell::\neg\neg A}{\ell::A} \qquad (l\wedge)\frac{\ell::A\wedge B}{\begin{array}{c}\ell::A\\ \ell::B\end{array}} \qquad (l\vee)\frac{\ell::\neg(A\wedge B)}{\ell::\neg A|\ell::\neg B} \quad .$$

For example, if the (l¬) rule is applied to the tableau that contains the formula 1::¬¬P the tableau can be extended by the formula 1::P.

5.2.2 Tableau Rules for LCKT4

We define the following ν and π rules for LCKT4:

ν-rules:

$$(lK)\frac{\ell::\Box A}{\ell.n::A}$$

For example, if the formula 1::□P appears on the tableau the tableau can be extended by the formula 1.2::P, provided 1.2 is a label that already appears on the branch.

$$(lT)\frac{\ell::\Box A}{\ell::A}$$

For example, if the formula 3::□(P ∧ Q) appears on the tableau we can extend the tableau by the formula 3::(P ∧ Q) after the application of the *(lT)* rule.

$$(l4)\frac{\ell::\Box A}{\ell.n::\Box A}$$

For instance, if this rule is applied to the formula 2::□Q the tableau will be extended by the formula 2.3::□Q, provided 2.3 is a label that already appears on the branch.

π-rule:

$$(l\pi)\frac{\ell::\neg\Box A}{\ell.n::\neg A}$$

where ℓ.n is new to the current branch.

For instance, if the formula 1::¬□(P ∧ ¬Q) appears on the tableau the tableau can be extended by the formula 1.2::¬(P ∧ ¬Q), after the application of the π -rule. Also note that in this case the prefix 1.2 will be new to the branch on which the formula ¬(P ∧ ¬Q) appears.

5.2.3 *Tableau Rules for LCKT5*

The following rules are defined for LCKT5:

ν-rules: LCPC

(*lK*)

(*lT*)

(*l4*)

$$(l4^r)\frac{\ell.n{::}\Box A}{\ell{::}\Box A}$$

For example, if the (*l*4^r) rule is applied to the formula 1.2::□P then the tableau can be extended by the formula 1::□P.

π-rules (*l*π)

5.3 Soundness

Theorem *65 (LC Soundness)*

If the explicit tableau for {¬A} is closed then A is valid in the corresponding classes of standard models of L.

Proof: It follows as a corollary of the following: If the systematic tableau for Γ, a finite set of formulas, closes then Γ is L-unsatisfiable (see definition 21, i.e. there is no L-model (where L denotes either KT4, KT4L or KT5) that satisfies Γ*)*. Refer to Gorḗ (1995:83) for details. □

5.4 Completeness

A systematic tableau is constructed by means of the construction algorithm as defined by Gorḗ (1995:78). This procedure is fundamentally the same as that described by Fitting except for a number of differences in the application of the ν and π rules as well as marking the completion of the formula. For comparative purposes we repeat Gorḗ's (1995) procedure here.

For the purpose of this algorithm a formula can be marked *awake*, *asleep* or *finished*. By *awake* we mean that the formula is available for use, that is, only awake formulas can be used in the application of the rules. By *asleep* we mean a formula that has been used in a rule and put to sleep until awakened by another rule. By *finished* we mean that a formula can never be used again in the application of a rule. Note that these markings do not impact on the closure of a tableau. In other words, if two prefixed formula, say $1.2::A$ and $1.2::\neg A$ occur on the same branch the branch is closed regardless of the formulas' markings.

For the sake of convenience the procedure UPDATE BRANCH ($\ell::A$) has been created below, where $\ell::A$ denotes any formula that needs to be added to the branch. Assume also that the set $\Gamma = \{A_1, A_2, \ldots A_n\}$ is the set of formulas for which we would like to create an explicit tableau.

```
Stage 1:      Put the labelled formulas $1::A_i$ where $A_i \in \Gamma$ in a vertical linear sequence of nodes, one
              beneath the other, in some order and mark them all as awake.

              n=1.
              WHILE the tableau is open and some formula is awake DO
Stage n+1       Choose an awake labelled formula $\ell::A$ as close to the root as possible.
(n ≥ 1)
                IF there are several awake formula at the same level THEN
                  Choose the one on the leftmost branch
                END IF

                IF $\ell::A$ is atomic THEN
                  Mark this formula as finished
                  Stop stage n + 1
                END IF

                 FOR each open branch B which passes through $\ell::A$ DO

                  IF $\ell::A$ is of the form $\ell::P \wedge Q$ THEN
                    UPDATE BRANCH ($\ell::P$)
                    UPDATE BRANCH ($\ell::Q$)
                  END IF


                  IF $\ell::A$ is of the form $\ell:: \neg(P \wedge Q)$ THEN
```

```
      Split B
      IF left branch THEN
        UPDATE BRANCH ($\ell::\neg P$)
      END IF
      IF right branch then
        UPDATE BRANCH ($\ell::\neg Q$)
      END IF
    END IF

    IF $\ell::A$ is of the form $\ell::\neg\neg P$ THEN
      UPDATE BRANCH ($\ell::P$)
    END IF

    IF $\ell::A$ is of the form $\ell::\Box P$ THEN
      FOR each applicable $\nu$ rule in LCL DO
        UPDATE BRANCH (corresponding denominator)
      END FOR
    END IF

    IF $\ell::A$ is of the form $\neg\Box P$ THEN
      k = the smallest integer such that the label $\ell k$ is new on the branch
      UPDATE BRANCH ($\ell k::\neg P$)
      Mark all formulas on B of type $\ell::\Box A$ awake
    END IF

  END FOR

  IF $\ell::A$ is of the form $\ell::\Box P$ THEN
    Mark $\ell::A$ as asleep
  ELSE
    Mark $\ell::A$ as finished
  END IF

n = n +1
END WHILE

UPDATE BRANCH ($\ell::A$)
```

```
        IF $\ell$::A does not already appear on the branch (with any mark) THEN
            Add the formula $\ell$::A to the end of the branch
            Mark $\ell$::A as awake
        END IF
    END UPDATE BRANCH
```

Notice that this systematic procedure constructs only one tableau and it traverses this tableau in a breadth-first manner.

For the sake of convenience, validity as defined in definition 23, is generalised here to *L-validity* where L denotes either KT4, KT4L or KT5. For example A is KT4-valid in a class C of models, written $\square_C$A, iff for every KT4 model M in C, $\square^M$A. Similarly, a formula A is said to be *L-satisfiable* (where L is either KT4, KT4L or KT5) if there exists a L-model M=<W,R,V> that satisfies (see definition 21) A.

Theorem *66 (LC Completeness)*

If A is L-valid then the systematic tableau for {¬A} must be closed.

Proof: The proof follows from the following result: If the systematic tableau for Γ does not close then Γ is L-satisfiable. (See Goré 1995:89 for details.)□

5.5 Applications

Goré's (1995) explicit tableau system is demonstrated below. The first example is for the KT4 system and the last demonstrates the explicit tableau system for KT5.

For KT4 we show that the formula ¬(□P ∧ ¬□□P) is valid in the class of models corresponding to KT4. From soundness we know that if we negate the formula and obtain a closed tableau then this formula is valid in the corresponding classes of standard models. In the following proof the right hand columns show the stages and the rules that were applied. Note that *a* denotes awake, *s* asleep and *f* finished. The column in which the *a, s* or *f* appears is the stage at which the formula was marked. For example 1::P was created during stage 4 by using the (*lT*) rule on the formula 1::□P and finished during stage 7.

Tableau	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**Rules**
1::¬¬(□P ∧ ¬□□P)	a	f										
1::(□P ∧ ¬□□P)		a	f									*l*¬
1::□P			a	s	a	s		a	s			*l*∧
1::¬□□P			a		f							*l*∧
1::P				a			f					*lT*
1.2::¬□P					a			f				*l*π
1.2::P						a				f		*lK*
1.2::□P						a					s	*l4*
1.2.3::¬P								a				*l*π
1.2.3::P											a	*lK*
1.2.3::□P											a	*l4*

Since both 1.2.3::¬P and 1.2.3::P appear on the same branch the branch is closed. This tableau only has one branch therefore the tableau is closed and we can conclude that the formula ¬(□P ∧ ¬□□P) is valid in the class of standard models that correspond to KT4.

In this example we show that the formula ¬(¬□¬P ∧ ¬□¬□¬P) is valid in the class of standard models that correspond to KT5. As described in the example above we negate the formula and the proof follows:

Tableau	1	2	3	4	5	6	7	8	9	10	11	Rules
1::¬¬(¬□¬P ∧ ¬□¬□¬P)	a	f										
|												
1::¬□¬P ∧ ¬□¬□¬P		a	f									$I^{\neg}$
|												
1::¬□¬P			a	f								$I^{\wedge}$
|												
1::¬□¬□¬P			a		f							$I^{\wedge}$
|												
1.2::¬¬P				a		f						I^{π}
|												
1.3::¬¬□¬P					a		f					I^{π}
|												
1.2::P						a		f				$I^{\neg}$
|												
1.3::□¬P							a		s			$I^{\neg}$
|												
1.3::¬P										f		*IT*
|												
1::□¬P									a		s	$I4^{r}$
|												
1.2::¬P											a	*IK*
|												
1::¬P											a	*IT*
|												
1.2::□¬P											a	*I4*

Here both 1.2::A and 1.2::¬A appear on the same branch as a result the tableau is closed and it follows that the initial formula is valid in the class of standard models that correspond to KT5.

5.6 Decidability

Showing decidability for Gorė' s (1995) procedure is similar to that of Fitting (1983) described in the section 3. If the reader was to compare the two procedures the reader will notice that the first set of alterations have already been implemented into Gorė's (1995) procedure. Therefore, the only alterations that are required are the changes that accommodate the logics that involve transitivity. The technical details are omitted here and the interested reader is referred to Gorė's (1995) work for the details.

6. CONCLUSION

In conclusion we provide a brief comparative study between the two techniques covered in this chapter. Both Fitting (1983) and Gorė (1995) provide systems for implicit and explicit tableau systems. Both systems use more or less the same tableau expansion rules, with Fitting presenting his tableau rules in terms of formula types such as π and ν and Gorė (1995) explicitly defining the rules. Note also that although these systems are fundamentally the same Gorė (1995) works with *sets* of formulas and his tableau expansion rules therefore incorporate these sets. Fitting's (1983) rules only act on one formula at a time.

As far as the systematic procedures are concerned Gorė (1995) differs from Fitting in that he implements some of the decidable features mentioned by Fitting (1983) directly into his procedure. For instance see the differences in the application of the ν and π rules. As a result Gorė (1995) appears to have a more efficient application than Fitting as demonstrated in the examples for KT4 above.

An additional feature of Gorė's (1995) system is that he incorporates a vast number of modal logics that Fitting (1983) did not initially include in his work. In particular he provides an expose for KT4L which Fitting did not do.

In lieu of the implementation of these systems the interested reader may want to note that Fitting (1988) has already provided implementation procedures for some of his tableau proof systems. In particular he has provided a tableau system for K4 using Prolog, which is essentially a tableau system for K with K4 embedded by means of simple mechanisations. He argues that to embed the other logics such as KT4, would require a similar exercise.

As noted at the start of this chapter Fitting (1983) and Gor$_{\text{é}}$ (1995) are not the only researchers that have tried to implement modal tableau systems using a variety of approaches that incorporate at least the logics KT4, KT4L and KT5. Laurent Catach (1991), for instance, developed a tableau system which he called TABLEAU. This system was initially implemented in VM/Prolog and appears to accommodate KT4, KT4L and KT5 as well as a number of other systems.

This chapter concludes the survey of tableau and resolution proof systems for propositional modal logic. In the next chapter we will demonstrate of how these tableau and resolution proof systems, initially defined for the normal systems of modal logic KT4, KT4L and KT5 can be applied to other branches of artificial intelligence, such as nonmonotonic logic.

CHAPTER 6

APPLICATIONS

What is laid down, ordered, factual, is never enough to embrace the whole truth: life always spills over the rim of every cup.

- Boris Pasternak

1. INTRODUCTION

The previous three chapters have been concerned with the specific modal logics KT4, KT4L and KT5. These systems were defined and an overview of the resolution and tableau systems that are available for these logics was provided. Chapter 6 addresses some applications of these logics in other branches of artificial intelligence.

The normal systems of modal logic KT4, KT4L and KT5 have properties that are similar to other logics in artificial intelligence, in particular nonmonotonic logic. This chapter will focus will on some of the applications of these systems in nonmonotonic logic. The main emphasis being on the relationships between KT4 and preferential models, KT4L and ranked preferential models and KT5 and autoepistemic logic. As a result of these relationships it should be clear that the theorem provers described in the previous chapters can be extended or applied in these other branches of logic as well.

2. NONMONOTONIC LOGIC

Nonmonotonic logic is essentially the study of the different ways that we can make inferences from information that does not satisfy the monotonicity property, denoted as:

$$\frac{A \models B, B \models C}{A \models C}$$

Where the relation □ is read as 'logically entails'.

For instance, if we were given the following premises:

- Garfield eats most birds.
- Garfield does not eat ostriches
- Tweety is a bird

We would probably conclude that Garfield will have Tweety for lunch. However, if we added the premise that *Tweety is an ostrich* we would probably want to conclude, without retracting our initial premises, that Garfield will have to make alternative arrangements for lunch. Applying monotonicity would have lead to the contradiction: Garfield eats Tweety and Garfield does not eat Tweety.

Nonmonotonic systems generally have at least the following components (Kraus, Lehmann & Magidor 1990):

1. A set of premises that describes the facts (e.g. cats are mammals) and definitions (e.g. child = ¬adult) of the system.
2. A set of conditional assertions which describes what we normally believe about the world.
3. Information of the situation at hand.

It is the second component that is of interest to us in this chapter. That is, the set of conditional assertions.

3. APPLICATIONS OF KT4

3.1 Preferential Models

In this section we introduce preferential models as discussed by Meyer, Labuschagne and Heidema (1996). In the next section we show how preferential models can be converted into KT4 models.

Preferential models are essentially models that give a model-theoretic account of the way one performs nonmonotonic inferences. The main idea being a partial ordering on possible states of the world (Lehmann & Magidor 1992). The nonmonotonic inferences in these models are based on the binary relation $|\!\sim$ which we define as follows (Meyer, Labuschagne & Heidema 1996:3):

Definition 87

Let $|\!\sim$ be a binary relation on a classical propositional language, (denoted by L_{PL}). $|\!\sim$ ***is a preferential consequence relation*** *iff it is closed under the following properties for every formula A, B, $C \in L_{PL}$.*

Reflexivity (Ref): $A \,|\!\sim A$

Left Logical Equivalence(LLE): *If A and B are logically equivalent and $A \,|\!\sim C$, then $B \,|\!\sim C$.*

Right Weakening (RW): *If B logically entails C and $A \,|\!\sim B$ then, then $A \,|\!\sim C$.*

Cautious Monotonicity (CM): *If $A \,|\!\sim B$ and $A \,|\!\sim C$, then $A \wedge B \,|\!\sim C$.*

And: *If $A \mid\sim B$ and $A \mid\sim C$, then $A \mid\sim B \wedge C$.*

Or: *If $A \mid\sim C$ and $B \mid\sim C$, then $A \vee B \mid\sim C$.*

Note that we read $A \mid\sim B$ as 'In the context provided by A, B is plausible'. (Meyer, Labuschagne & Heidema 1996). Before we define a preferential model we require the following definitions.

Definition 88

The triple $P=(S, L, R)$ is called a ***model*** *where S, the set of states, is any set, L the labelling function, is a function from S to I (the set of all interpretations of L_{PL}) and R is a binary relation on S. The set of interpretations satisfying a formula A is denoted by $\hat{A}$. That is, for every $A \in L_{PL}$, $\hat{A} = \{s \mid I = L(s)$ and $I(A) = T\}$ (Meyer, Labuschagne & Heidema 1996:3).*

Definition 89

Let R be a relation on a set Γ, and let Γ_s be any subset of Γ. An element $A \in \Gamma_s$ is ***minimal*** *in Γ_s iff for every $B \in \Gamma_s$, $(B, A) \notin R$ (Meyer, Labuschagne & Heidema 1996:3).*

Definition 90

Let R be a relation on a set Γ, and let Γ_s be any subset of Γ. Γ_s is defined to be ***smooth*** *iff for every $A \in \Gamma_s$, either A is minimal in Γ_s, or there is a minimal $B \in \Gamma_s$ such that $(B, A) \in R$ (Meyer, Labuschagne & Heidema 1996:3).*

Preferential models are defined as follows (Meyer, Labuschagne & Heidema 1996:4):

Definition 91

Let $P = \langle S, L, \prec \rangle$ be a model with a strict partial order (irreflexive and transitive) on S. P defines a relation $\mid\sim_P$ on L_{PL} as follows: $A \mid\sim_P B$ iff for every S minimal in $\hat{A}$, $S \in \hat{B}$, for any formula A and B. P satisfies the smoothness condition iff for every $A \in L_{PL}$, $\hat{A}$ is smooth. If P satisfies the smoothness condition, it is called a ***preferential model****.*

Alternatively a preferential model $P = \langle S, L, \prec \rangle$ is a triple where (Boutilier 1990):

- S = a set of possible worlds.
- L maps propositional variables into 2^S, i.e. L(A) is the set of worlds where A holds.
- $\prec$ is a strict partial order on S such that for all propositional formula A, $\hat{A}$ is smooth.

We now establish the relationship between preferential consequence relations and preferential models.

Theorem 67

Every preferential model P defines a preferential consequence relation. Conversely, for every preferential consequence relation $|\sim$ *there is a preferential model P such that* $|\sim\ =\ |\sim_P$.

(See Kraus, Lehmann & Magidor 1990 for proof.)

We stop here and return to modal logic. We refer the interested reader to the work of Boutilier (1994), Meyer, Labuschagne & Heidema (1996), Kraus, Lehmann and Magidor (1990) and de Rijke (1996) for more information with regard to preferential models.

3.2 Relation With Modal Logic

Provided we are only interested in the relation $|\sim_P$, it is easy to see that preferential models P = <S, L, $\prec$> are essentially the set of KT4-models. As a result these preferential models can be converted to KT4 models, demonstrated as follows:

Let P = <S, L, $\prec$> be a preferential model. We want to translate P to a KT4 model M=<W,R,V> where R is reflexive and transitive. So let W = S. From the previous section we noted that the relation $\prec$ is a strict partial order and we know from chapter 3 that R is a pre-order relation (i.e. reflexive and transitive). We build R from $\prec$ as follows: $R = \prec \cup \{(\alpha,\alpha) \mid \alpha \in S\}$. Lastly we need to define V for M. From the above definition of L we have that for any atom A, A is true at α iff $L(\alpha) = I$ and $I(A) = T$. We use this result to create the mapping V(A) for any A. That is, let V(A) be the set $\{\alpha \mid L(\alpha) = I$ and $I(A) = T\}$ then V(A) will contain all the worlds at which A is true. This concludes the translation. To summarise, we have a KT4 model M = <W,R,V> where :

- W = S
- $R = \prec \cup \{(\alpha,\alpha) \mid \alpha \in S\}$ and
- $V(A) = \{\alpha \mid L(\alpha) = I$ and $I(A) = T\}$ for any atom A.

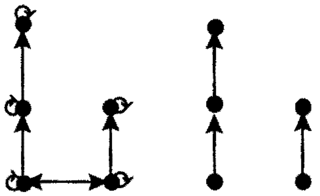
Conversely we can convert a KT4 model = <W, R, V> to a preferential model $P = (S, L, \prec)$ model as follows:

- $S = W$

- For any $\alpha \in S$ we define I as follows: for every atom P, $I(P) = T$ if and only if $\alpha \in V(P)$. Now set $L(\alpha) = I$.

- $\prec = R - (\{(\alpha,\alpha) \mid \alpha R\alpha\} \cup \{(\alpha,\beta) \mid \alpha R\beta \text{ and } \beta R\alpha\})$. That is, we remove all reflexive pairs from the model and if there are any symmetric pairs we remove both symmetric pairs as well.

For instance, the model on the left is a reflexive transitive model and the model on the right is the corresponding strict partial order model.



Clearly just being able to convert a preferential model P to a KT4 model is of no value if we cannot use KT4 to derive any results about plausibility as defined above. That is, ideally we would like to define a preferential consequence relation on KT4 that will allow us to derive results about plausibility in the same way the relation |~ does.

Now if we were to define a binary relation ⊳ on propositional well formed formulas such that A ⊳ B iff $\Box^M A \rightarrow \Diamond(A \wedge \Box(A \rightarrow B))$ where A and B are any propositional well formed formulas and M=<W, R, V> is any KT4 model, we would have the desired preferential consequence relation defined for KT4.

Theorem 68

Every binary relation ⊳ obtained from a KT4 model M is a preferential consequence relation. Conversely, for every preferential consequence relation |~ there is a KT4 model M such that A |~ B iff A ⊳ B.

(See Meyer, Labuschagne & Heidema 1996 for a proof of a similar result, or refer to Kraus, Lehmann & Magidor 1990 for more details.)

The advantage of being able to define plausibility using KT4 models is that now we are in a position to prove results in preferential models by using the resolution and tableau proof systems for KT4 as discussed in the previous chapters.

4. APPLICATIONS OF KT4L

4.1 Ranked preferential Models

Any good nonmonotonic reasoning system will validate all the inference rules for a preferential model. However, there are certain principles of reasoning that are not supported by these models. Three of these principles are:

Negation Rationality: If $A \mid\!\sim B$ then $A \wedge C \mid\!\sim B$ or $A \wedge \neg C \mid\!\sim B$.

Disjunctive Rationality: If $A \vee B \mid\!\sim C$ then $A \mid\!\sim C$ or $B \mid\!\sim C$.

Rational Monotonicity: If $A \mid\!\sim C$ then $A \wedge B \mid\!\sim C$ or $A \mid\!\sim \neg B$.

Rational Monotonicity is the strongest principle of the three given. That is, if we have rational monotonicity then we have negation and disjunctive rationality. Lehmann & Magidor (1992) argue that any reasonable nonmonotonic inference procedure should, over and above the rules defined for preferential models, also define a rational relation. They use the following example to justify this: For any propositional atom P and Q, if we have $P \mid\!\sim Q$ we would expect $P \wedge R \mid\!\sim Q$ to follow from a knowledge base where R is a new assertion such that $R \neq P$ and $R \neq Q$, since we have no information about the influence of R on the objects of P and Q it would be sensible to assume that there was no influence and that normal $P \wedge R$ objects are just P objects. It follows, therefore, that it would be convenient if the relation $\mid\!\sim$ had the property of rational monotonicity. Relations represented by a ranked preferential models have this desired property. In other words, a new consequence relation can be defined that performs the same way as the preferential consequence relation except that it is also rational.

Definition 92

*A **rational consequence** relation is a preferential consequence relation that satisfies rational monotonicity (Boutilier 1990:123).*

Definition 93

*$(X, \leq)$ is a **totally ordered set** iff for every $\alpha \in X$ and $\beta \in X$ either α is accessible to β or β is accessible to α denoted as $\alpha \leq \beta$ or $\beta \leq \alpha$.*

Definition **94**

A ***ranked*** *model (or a K-model) is a preferential model $K = <S, L, \prec>$ where the relation $\prec$ is such that there exists a totally ordered set $(X, <)$ and a function $f: S \rightarrow X$, where $s \prec t$ iff $f(s) < f(t)$ (i.e. $f(s) \leq f(t)$ and $f(s) \neq f(t)$) (Boutilier 1990:123).*

Theorem *69*

$|\sim$ is a preferential consequence relation that satisfies rational monotonicity iff it is the consequence relation defined by some K-model.

(See Boutilier 1990:124 for proof.)

Note that a preferential consequence relation that satisfies rational monotonicity is called a rational consequence relation.

4.2 Relation to Modal Logic

Preferential ranked models are strict cases of KT4L models. That is, as with KT4 and preferential models above, we can also show that preferential ranked models can be converted into KT4L models and vice versa.

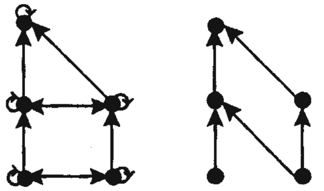
Consider the preferential ranked model $K = <S, L, \prec)$. We can convert this model into a KT4L model $M=<W, R, V>$ as follows:

- $W = S$
- $R = \prec \cup \{(\alpha,\beta) \mid \alpha \not\prec \beta \text{ and } \beta \not\prec \alpha\}$
- $V(A) = \{\alpha \mid L(\alpha) = I \text{ and } I(A) = T\}$ for every atom A.

Conversely we can convert a KT4L model $= <W, R, V>$ to a $K = (S, L, \prec)$ model as follows:

- $S = W$
- For any $\alpha \in S$ we define I as follows: for every atom P, $I(P) = T$ if and only if $\alpha \in V(P)$. Now set $L(\alpha) = I$.
- $\prec = R - \{(\alpha,\beta) \mid \alpha R\beta \text{ and } \beta R\alpha\}$. That is, we remove all reflexive pairs from the model and any horizontal connections, in order to ensure that models on the same level are not comparable.

For example, in the following diagram the model on the left is a KT4L model and the model on the right is its corresponding K-model.



As with KT4 it is possible to define a rational consequence relation for KT4L. Now if we define the relation $\rhd$ such that only those formulas that satisfy the condition $\Box^{M} A \rightarrow \Diamond(A \rightarrow \Box(A \rightarrow B))$ satisfy the relation $\rhd$ we would have a rational consequence relation for KT4L. That is, if $A \rhd B$ iff $\Box^{M} A \rightarrow \Diamond(A \rightarrow \Box(A \rightarrow B))$ for every formula A and B then $\rhd$ is a rational consequence relation.

Theorem 70

Every binary relation $\rhd$ obtained from a KT4L model M defines a rational consequence relation. Conversely, for every rational consequence relation $\mid\sim$ there is a KT4L model M such that $A \mid\sim B$ iff $A \rhd B$.

Since it is possible to translate ranked preferential models into KT4L we now have the benefit of deriving plausibility results for ranked preferential models using KT4L resolution and tableau proof systems.

5. APPLICATIONS OF KT5

5.1 Autoepistemic Logic

Autoepistemic logic is another addition to non-monotonic logic that has applications within the realm of modal logic. Technically, autoepistemic reasoning is the drawing of conclusions in the absence of some specific information. For instance, consider the following formula:

$$\forall x\ Cat(x) \wedge Mhas\text{-}Parents(x) \rightarrow Has\text{-}Parents(x).$$

If we interpret M as 'it is consistent to believe', then the following reading is apparent: 'For each x, if x is a cat and it is consistent to believe that x has parents, then x has parents.' To accept this reading, from an autoepistemic perspective, we must believe that we know all the instances of cats that have no parents. That is, if we infer that Garfield has parents, then we are basing our observation on the subjective opinion that we know all the instances of cats who have no parents. (Lukaszewicz 1990.)

In the remainder of this section and the next we will provide a brief overview of autoepistemic logic and its applications to modal logic. This exposition will be based on the work of Witold Lukaszewicz (1990) and the reader is therefore referred to his work for more details with regards to this subject.

For the purpose of this thesis the focus will be on propositional autoepistemic logic. In this language the □ that we have used previously is now read as 'it is believed' and takes the place of the M in the above example. The ◊ becomes ¬□¬ which is similar to the work we have discussed in previous chapters. The remainder of the formulas in this language are similar to those defined in chapter 3 with the exception that the focus in this section is on a specific set of formulas, referred to as *belief sets*. These sets are viewed as the total collection of beliefs of an agent reasoning about its own beliefs. (Lukaszewicz 1990.)

As with modal logic, model-theoretic semantics can also be defined for autoepistemic logic. That is, we can define interpretations, truth values of formulas within these interpretations and models, for autoepistemic logic. We do so in terms of the following definitions.

Definition 95

*An **autoepistemic interpretation** (AE interpretation) for L (the propositional language) is a pair V = <m, S>, where m is an assignment of truth-values to the atoms occurring in L^* and $S \subseteq L$ is a belief set. Such an interpretation will usually be referred to as an autoepistemic interpretation of S (Lukaszewicz 1990:124).*

In other words an AE interpretation <m, S> is essentially a description of a particular world, together with an agent situated in it. That is, m specifies what is actually true in the world, whereas S determines what the agent actually believes. (Lukaszewicz 1990:124.)

Definition 96

Let V=<m, S> be an AE interpretation. The ***value of a formula*** *A in V, denoted by $V(A)$, is a truth-value specified by the following rules (where T denotes 'true' and F denotes 'false'):*

1. $V(P) = m(P)$ *for any atom P;*
2. $V(\neg B) = T$ *iff $V(B) = F$;*
3. $V(B \rightarrow C) = T$ *iff $V(B) = F$ or $V(C) = T$;*
4. $V(B \vee C) = T$ *iff $V(B) = T$ or $V(C) = T$;*
5. $V(B \wedge C) = T$ *iff $V(B) = T$ and $V(C) = T$;*
6. $V(\square B) = T$ *iff $B \in S$.*

Definition 97

Let V=<m, S> be an AE interpretation of S and suppose that Γ is a set of formulas. V is an ***autoepistemic model*** *(AE model) of Γ iff $V(A) = T$, for each $A \in \Gamma$. We shall write $\Gamma \square_S A$ to indicate that a formula A is true in every AE interpretation of S which is an AE model of Γ (Lukaszewicz 1990:124).*

Naturally, as with any other logic, inferences are also possible in autoepistemic logic. What these inferences are or how they take place are not within the scope of this thesis. However, we will show that an initial set of premises or beliefs can be extended to a set of beliefs that any ideal rational agent would accept as the basis of any of its inferences. In other words, if we define the set T as the initial set of premises, generally referred to as a theory, we can extend T to a set of beliefs S of an ideally rational agent.

Lukaszewicz (1990) argues that according to Moore there are at least two constraints that have to be imposed on a belief set S of an ideally rational agent reasoning on the basis of its set of premises. These constraints are *soundness* and *semantic completeness* which we define below.

Definition 98

A belief set S is ***sound*** *with respect to a theory T (set of initial premises) if and only if every AE interpretation of S which is an AE model of T is also an AE model of S (Lukaszewicz 1990:124).*

Roughly speaking from soundness we can infer that if all things in T are true then all things in S are also true.

Definition 99

A belief set S is ***semantically complete*** *if and only if S contains any formula A, provided that A is true in every AE interpretation of S which is an AE model of S (Lukaszewicz 1990:125).*

That is, in any AE interpretation V = <m, S>, if A is true in V and all the formulas in S are true in V then A $\in$ S.

The belief set S of an ideally rational agent can now be defined in terms of soundness, semantic completeness and the initial set of premises T (Lukaszewicz 1990:125).

Definition 100

A belief set S is an ***AE extension*** *of a theory (set of premises) T iff*

1. S is sound with respect to T;

2. S is semantically complete;

3. $T \subseteq S$.

We conclude this section and move to the next section where we will show that an AE extension S of T (as defined above) can be constructed in terms of a KT5 frame.

5.2 Relation to Modal Logic

Model-theoretic semantics can be defined for autoepistemic logic in terms of KT5 frames.

Definition 101

A complete ***KT5-frame*** *for L (some language) is a pair <W, m>, where W is a set of possible worlds and m is a function which assigns to each pair, consisting of a proposition constant from L and an element of W, an element of {T,F} (Lukaszewicz 1990:133).*

For instance, m(α,P)=T means that the atom P is true at the world α. In other words a KT5-frame is essentially a universal KT5-model M = <W, R, V> where m takes the place of V and R is the universal relation on W, i.e. R = W x W.

Truth in a KT5-frame is usually defined in terms of specific interpretations.

Definition 102

*A **possible-world AE interpretation** for L is a pair PV=<m_{PV}, M>, where m_{PV} is an assignment of truth values to the atoms of L and M is a complete KT5 frame for L. We say that PV is a **possible-world interpretation** of S, if S is the set of all formulas which are true in M (Lukaszewicz 1990:134).*

To evaluate a formula in this interpretation we require a definition similar to that of definition 96 (Lukaszewicz 1990:134):

Definition 103

*Let PV=<m_{PV}, M> be a possible-world AE interpretation. The **value of A in PV** denoted PV(A), is a truth-value defined by the following rules:*

1. $PV(P) = m_{PV}(P)$ *for any atom P;*
2. $PV(\neg B) = T$ *iff* $PV(B) = F$;
3. $PV(B \rightarrow C) = T$ *iff* $PV(B) = F$ *or* $PV(C) = T$;
4. $PV(\square B) = T$ *iff* $\square B$ *is true in M.*

Definition 104

*A possible-world AE interpretation of $S \subseteq L$ in which all the formulas of Γ (a set of formulas) $\subseteq L$ are true is called a **possible-world AE mode** of Γ (Lukaszewicz 1990:134).*

From the above semantics we can show that there exists a set S, defined in terms of KT5 frames, that is an AE extension of T (defined previously).

Theorem 71

S is an AE extension of T if and only if $S = \{A \mid T \cup \square S_0 \cup \neg\square \bar{S}_0 \square_{PV} A\}$ *where:*

S_0 = $\{A \in S \mid A \text{ has modal degree of zero}\}$

$\square S_0$ = $\{\square A \mid A \in S_0\}$

$\bar{S}_0$ = $\{A \mid A \notin S_0\}$

$\neg\square \bar{S}_0$ = $\{\neg\square A \mid A \notin S_0\}$

$\Gamma \square_{PV} A$ = For all models PV in which all the formulas of Γ are true, A is also true. That is, for every possible-world AE interpretation PV in which all well formed formulas in Γ are true, we have that A is true in PV.

(See Lukaszewicz 1990:139 for proof.)

In conclusion note that there is an interesting relationship between autoepistemic logic and K45 modal logic.

The modal system K45 is the system KT5 without the axiom schema T: $\square A \rightarrow A$. That is, the appropriate models for K45 are those in which the accessibility relation is transitive and euclidean (i.e. if $M=<W, R, V>$ then for any α, β, and $\gamma \in W$, if $\alpha R \beta$ and $\alpha R \gamma$ then $\beta R \gamma$).

To demonstrate the relationship between K45 modal logic and autoepistemic logic we require the following definition.

Definition 105

We say that a formula A is ***strongly K45-provable*** *from a theory T, written* $T \square^{s}_{K45} A$ *iff there are formulas* $A_1, A_2, \ldots A_n \in T$ *such that* $\square_{K45} A_1 \wedge A_2 \wedge \ldots \wedge An \rightarrow A$ *(Lukaszewicz 1990:138).*

It can be shown that the strong K45-provability relation is precisely the syntactic counterpart of the relation $\square_{PV}$.

Theorem 72

For any $T \subseteq L_T$ *(language of a theory T) and any* $A \in L_T$ *:* $T \square_{PV} A$ *if and only if* $T \square^{s}_{K45} A$*.*

(See Konolige 1988 for proof.)

From the discussion above it should be clear that it is possible to translate an autoepistemic logic into an equivalent modal logic and vice versa. This in turn means that we are able to use any of the resolution and tableau systems described in chapter 4 and 5 for KT5, in certain aspects of autoepistemic logic.

6. CONCLUSION

A brief review of what was covered in this thesis is supplied here starting with chapter 1 which provided a brief historical exposition of propositional modal logic. Chapter 2 introduced propositional logic providing an introduction to both resolution and tableaux proof systems from a classical perspective. Chapter 3 introduced propositional modal logic as well as the three normal systems of modal logic KT4, KT4L and KT5. Chapter 4 focused on resolution proof systems for propositional modal logic, providing details on how to use these systems in order to determine local or global logical entailment. The rules applicable to these systems, soundness and completeness results and applications of these systems to the normal systems of modal logic KT4, KT4L and KT5 were also supplied. Chapter 5 on the other hand, focused primarily on tableaux proof systems for propositional modal logic. Soundness, completeness and decidability results where provided for these systems. Applications of these systems were demonstrated for the normal systems of modal logic KT4, KT4L and KT5. Chapter 6 concluded this thesis with some suggestions for the applications of KT4, KT4L and KT5 to other branches of logic, in particular nonmonotonic logic. Relationships between KT4 and preferential models, KT4L and ranked preferential models and KT5 and autoepistemic models were also illustrated.

The writer hopes that the reader found this summary of the relevant resolution and tableau systems that are available to modal logic and their respective applications to the logics KT4, KT4L and KT5 useful. The writer also hopes that this work will inspire readers to implement the different resolution and tableau proof systems, demonstrating that as with the other branches of logic, such as propositional logic and predicate logic, modal logic can also make beneficial contributions to the realms of artificial intelligence.

References

ABADI, M. & MANNA, Z. 1986. Modal Theorem Proving. *Lecture Notes in Computer Science*. Edited by J.H. Siekmann. Heidelberg: Springer-Verlag. 230, July:172-189.

AUFFRAY, Y & ENJALBERT, P. 1992. Modal Theorem Proving. An Equational Viewpoint. *Journal of Logic and Computation* 2(3):247-295.

BLACKBURN P., JASPARS J. & DE RIJKE, M. 1996. Reasoning about Changing Information. Lecture Notes, Winter School on Formal and Applied Computer Science, July 1-12, Cape Town: University of Cape Town.

BOUTILIER, C. 1994. Conditional Logics of Normality. A Modal Approach. *Artificial Intelligence*, 68:87-154.

CATACH, L. 1991. TABLEAUX. A General Theorem Prover for Modal Logics. *Journal of Automated Reasoning*, 7:489-510.

CHAN, M.C. 1985. An Introduction to the Recursive Resolution Method for Modal Logic. Technical Report No. 7/85. Department of Computer Science, La Trobe University, Melbourne, Australia.

CHAN, M.C. 1987. The Recursive Resolution Method for Modal Logic. *New Generation Computing*, 5:155-183.

CHANG, C. & LEE, R.C. 1973. *Symbolic Logic and Mechanical Theorem Proving*. New York and London : Academic Press.

CHELLAS, B.F. 1980. *Modal Logic*. An Introduction. Cambridge: Cambridge University Press.

DEL CERRO, L. & HERZIG, A. 1988. Linear Modal Deductions. *Lecture Notes in Computer Science*, Edited By G. Goos & J. Harmanis. Heidelberg: Springer-Verlag. 310:487-499.

FITTING, M. 1983. *Proof Methods for Modal and Intuitionistic Logics*. Dordrecht, Holland: D. Reidel Publishing Company.

FITTING, M. 1988. First-Order Modal Tableaux. *Journal of Automated Reasoning*, 4:191-213.

FITTING, M. 1990a. Destructive Modal Resolution. *Journal of Logic and Computation*, 1(1), July:83-97.

FITTING, M. 1990b. *First-Order Logic and Automated Theorem Proving*. New York: Springer-Verlag.

GENESERETH, M R. & NILSSON, N. J. 1987. *Logical Foundations of Artificial Intelligence*. Palo Alto: Morgan Kaufmann Publishers, Inc.

GENT, I.P. 1993. Analytic Proof Systems for Classical and Modal Logics of Restricted Quantification. Cs-Thesis-Gent93, Department of Computer Science, University of Warwick, Warwick, England.

GOR$_é$ R. 1995. Tableau Methods for Modal and Temporal Logics. Technical Report Tr-Arp-15-95. Australian National University, Australia.

HAMILTON, A.G. 1988. *Logic for Mathematicians*. Cambridge: Cambridge University Press.

HINTIKKA, K.J.J. 1955. Form and Content in Quantification Theory. *Acta Philosophica Fennica*, 8:3-55.

HUGHES, G.E. & CRESSWELL, M.J. 1996. *A New Introduction to Modal Logic*. London: Routledge.

KRAUS, S., LEHMANN, D. & MAGIDOR, M. 1989. Nonmonotonic Reasoning, Preferential Models and Cumulative Logics. *Artificial Intelligence*, 41:167-207.

LEHMANN, D. & MAGIDOR, M. 1992. What Does a Conditional Knowledge Base Entail? *Artificial Intelligence* 55:1-60.

LUKASZEWICZ, W. 1990. *Non-Monotonic Reasoning. Formalization of Commonsense Reasoning.* London: Ellis Horwood.

MEYER, T.A., LABUSCHAGNE, W.A. & HEIDEMA J. 1996. Conditional Plausibility by Power-Orders using S-Models. Research Report, 232/96(11), University of South Africa, Pretoria, South Africa.

OHLBACH, H.J. 1988. A Resolution Calculus for Modal Logic. *Lecture Notes in Computer Science*. Edited by G. Goos & J Hermanius. Heidelberg: Springer-Verlag. 310.

OPHELDERS, W.M.J & DE SWART, H.C.M. 1993. Tableaux versus Resolution a Comparison. *Fundamenta Informaticae* 18:109-12.

RAUTENBERG, W. 1983. Modal Tableau Calculi and Interpolation. *Journal of Philosophical Logic*, 12:403-423.

In ending, I want to express a hope that … [logic] may serve now as a model for the solution of the main problem of our epoch: to reveal a supreme religious goal and to fathom the meaning of the spiritual activity of mankind.
- Shafarevitch