ISSN 1361 - 6161

THE UNIVERSITY of MANCHESTER

Computer Science
University of Manchester

Decision procedures for Temporal Logic

Graham D Gough

Department of Computer Science

University of Manchester

Technical Report Series

UMCS-89-10-1

Decision procedures for Temporal Logic

Graham D Gough

Department of Computer Science
University of Manchester
Oxford Road, Manchester, UK.
graham@cs.man.ac.uk

October 1984

Abstract

Temporal Logic has been shown [MP82, MP83, BK83, BKP84], to be a powerful tool for reasoning about concurrent programs. A decision procedure is an algorithm for determining whether a given formula of the logic is valid (i.e. it is true in all interpretations of the logic). This dissertation is a study of decision procedures for various forms of temporal logic; in particular an improved decision procedure is given for for a future time Temporal Logic, and a new decision procedure is given for a form of Temporal Logic including past time operators.

Contents

Chapter 1

Introduction

Temporal Logic has been shown [MP82, MP83, BK83, BKP84], to be a powerful tool for reasoning about concurrent programs. A decision procedure is an algorithm for determining whether a given formula of the logic is valid (i.e. true in all interpretations of the logic).

In this dissertation an improved decision procedure is given for a future time Temporal Logic, which is used to show that an axiomatisation of the logic is complete.

Barringer and Kuiper [BK83] have developed a hierarchical specification method using Temporal Logic which, through the introduction of operators for describing past time requirements of execution sequences, handles both safety and liveness conditions. A new decision procedure is given for a Temporal Logic including such past time operators.

In this chapter we give a very brief introduction to modal and temporal logics and the use of temporal logic for reasoning about programs. For fuller accounts of these matters the reader is referred to [HC72, RU71] for the former and to [MP82, MP83, BAMP81] for the latter.

1.1 Modal Logics.

Modal logic has been described [HC72] as the logic of necessity and possibility, of "must be" and "may be". Systems of modal propositional logic, as well as having the normal apparatus of propositional calculus, are equipped with two additional operators called the necessity and possibility operators, denoted here by L and M respectively. Formulae such as $L(p \Rightarrow q)$ and $p \Rightarrow Mq$ are read as "it is necessary that p implies q" and "p implies that that q is possibly true". The symbols $\Box$ and $\Diamond$ are often used in place of L and M respectively.

The terms necessity and possibility are subject to a large number of semantic interpretations, for example moral, epistemic and temporal; but in all cases the semantics are essentially based on Leibniz's characterisation of necessity:

> A proposition is necessary if it is true in all possible worlds.

This, of course begs the definitions of both "worlds" and "possible", and the enormous variety of modal logics arises from the choices made for these definitions.

In propositional modal logic a world is simply an assignment of truth values to all propositions in the language. We assume that the logic contains the usual boolean connectives $\neg$, $\vee$, $\wedge$, etc. and that the assignment of truth values to formulae is done in a way that is consistent with classical truth tables.

A formula in a modal logic is a statement made in the context of a particular world, usually called the current world, or current state of affairs. The phrase "possible world" then means a world that is in some sense accessible from the current world.

So, to summarise, a modal system $\mathcal{S}$ consists of

i) A language, i.e. a set of well-formed formulae F, including the set of atomic propositions $Prop$.

ii) A universe, or set of worlds U, each being an an assignment of truth values to $Prop$.

iii) A binary relation R, called accessibility, on the set of worlds, U.

Given such a modal system, the formula Lu is true in a given world w, which we denote by $w \models Lu$, if and only if u is true in all worlds accessible from w, that is in all worlds w_1 such that $(w, w_1) \in R$. The formula Mu is true in w if there exists a world w_1 such that $(w, w_1) \in R$ in which u is true.

Thus modal logic is an appropriate tool for discussion of properties that change from one world, or state of affairs, to another, without specific reference to the worlds involved. The many different forms of modal logic are largely distinguished by the properties which the relation R is assumed to possess, see for example [HC72]. We now concentrate on the temporal forms of modal logic.

1.2 Temporal Logic.

Temporal Logics are forms of modal logic in which the worlds are interpreted as states of affairs at different times and the accessibility relation is an ordering on time, so that a world s_1 is accessible from a world s if s can change into s_1 through passage of time. The description "state" is more usual in this context than world. Different temporal systems reflect different views of the underlying structure of time, for example continuous or discrete, linear or branching.

The two forms of temporal logic which are most commonly used for reasoning about programs, Linear Time and Branching Time, share the assumption that time is discrete. This is not a decision taken on philosophical grounds regarding the nature of time, but an assumption justified by the computational model used when reasoning about programs. Since the accessibility relation is to be interpreted as an ordering on time we also assume that the relation R is

i) reflexive; so for all $s \in U$, $(s, s) \in R$, and

ii) transitive; so if $s, t, u \in U$ with $(s, t) \in R$ and $(t, u) \in R$, then $(s, u) \in R$.

We further assume the existence of a second relation, r, on U, called the immediate accessibility relation, such that R is the reflexive transitive closure of r. (If $(s, s_1) \in r$, s_1 is thought of as a next moment in time to s).

The two most commonly used forms of Temporal Logic, Branching Time and Linear Time, also share the assumption that time is backwards linear, that is,

> If $(s_1, s) \in R$ and $(s_2, s) \in R$ then either $(s_1, s_2) \in R$ or $(s_2, s_1) \in R$

This means that any state has a unique history.

1.2.1 Branching Time

If R is assumed to be transitive, reflexive, and backward linear, we obtain a model of time as a tree-like structure, which branches in a forward direction but is linear if we look backwards. Thus any state of affairs has a unique history but possibly many futures. This model is particularly useful where nondeterminism - beyond that used to model concurrency - is involved. This is the Branching Time model.

1.2.2 Linear Time

In a linear time temporal logic the relation R is not only backwards linear, but also forwards linear, so that

> if $(s, s_1) \in R$ and $(s, s_2) \in R$ then either $(s_1, s_2) \in R$ or $(s_2, s_1) \in R$

These properties imply that the immediate successor relation, r satisfies

> For all s, there is at most one s_1 with $(s, s_1) \in r$ and there is at most one s_2 with $(s_2, s) \in r$.

This means that a model in Linear Time temporal logic consists of a sequence of states, indexed by some subset of the integers, each having an assignment of the propositions of the logic. If we put no further restriction on R (or r) then the sequences can be finite or infinite; different models of time within the linear framework can be obtained by insisting, for example that

> For all s, there is exactly one s_1 with $(s, s_1) \in r$

which gives an infinite future, or

> For all s, there is exactly one s_1 with $(s_1, s) \in r$

which gives an infinite past, or

For some s, there is no s_1 with $(s, s_1) \in r$

which gives a finite future, or

For some s, there is no s_1 with $(s_1, s) \in r$

which gives a finite past,

1.3 Temporal logic and programs.

Linear time Temporal Logic was introduced by Pnueli [Pnu77] as a tool for reasoning about concurrent programs. Using a shared variable, interleaving model of concurrency the logic can be used to describe invariance and eventuality properties of a program's execution sequence without the explicit introduction of a variable describing time. Linear time temporal logic formulae are used to describe the properties required of all execution sequences. Details of such techniques can be found in [MP82, MP83].

The linear model of time, therefore, is appropriate for reasoning about universal properties of programs, those shared by all execution sequences. The Branching Time approach considers the set of all execution trees generated by a program and so can be used to reason about existential properties of concurrent programs; for example, showing that for a non-deterministic program there exists at least one execution sequence which terminates correctly.

1.4 Description of dissertation.

- Chapter 2 describes a Linear Time Temporal Logic using future time operators, $\mathcal{FL}$, and a decision procedure for the logic, based on semantic tableaux.
- Chapter 3 contains an axiomatisation for the logic $\mathcal{FL}$ and a proof, using the decision procedure, that the axiomatisation is complete.
- Chapter 4 introduces a Linear Time logic containing both future and past time operators, and gives a decision procedure for the logic.
- Chapter 5 considers the problems raised by introducing bounded time.
- Chapter 6 discusses the implementation of the algorithms discussed in earlier chapters.
- Chapter 7 describes and compares other decision procedures.

Chapter 2

The logic $\mathcal{FL}$.

The main purpose of this chapter is to describe the linear future time temporal logic $\mathcal{FL}$ together with a decision procedure for the logic. Many of the definitions and ideas introduced in this chapter will also be used in a later chapter when past-time operators are introduced.

2.1 Definition of the language.

The form of temporal logic used is essentially classical propositional logic, augmented by five temporal operators :

$\bigcirc$	nexttime,
$\mathcal{U}$	until,
$\square$	henceforth (or always in the future),
$\Diamond$	sometime (in the future),
$\mathcal{W}$	unless (or weak until).

More precisely we have

2.1.1 Syntax.

The language $\mathcal{FL}$ consists of

- A set Prop of atomic propositions.
- Logical constants **true**, **false**
- Boolean connectives $\wedge, \neg, \vee, \Rightarrow, =$
- Temporal operators $\bigcirc$, $\mathcal{U}$, $\square$, $\Diamond$, $\mathcal{W}$

with the following formation rules for formulae in $\mathcal{FL}$

- Any $p \in Prop$ is a formula.

- **true**, **false** are formulae.
- If f, g are formulae then so are

$$f \wedge g,\ \neg f,\ f \vee g,\ f \Rightarrow g,\ f \Leftrightarrow g,\ \bigcirc f,\ \Box f,\ \Diamond f,\ f\,\mathcal{U} g,\ f\,\mathcal{W} g.$$

- All formulae in $\mathcal{FL}$ are formed by the above rules.

Definition 1 *The* degree *of a formula f in $\mathcal{FL}$ is denoted by deg(f) and is defined as follows*

$$\begin{array}{rcl}
deg(p) & = & 1 \text{ for } p \in Prop \\
deg(\neg f) & = & deg(f)+1 \\
deg(f \wedge g) & = & deg(f)+deg(g)+1 \\
deg(f+g) & = & deg(f)+deg(g)+1 \\
deg(f \Rightarrow g) & = & deg(f)+deg(g)+1 \\
deg(f \Leftrightarrow g) & = & deg(f)+deg(g)+1 \\
deg(\bigcirc f) & = & deg(f)+1 \\
deg(\Box f) & = & deg(f)+2 \\
deg(\Diamond f) & = & deg(f)+2 \\
deg(f\,\mathcal{U} g) & = & deg(f)+deg(g)+2 \\
deg(f\,\mathcal{W} g) & = & deg(f)+deg(g)+2
\end{array}$$

(Note that, if $|f|$ denotes the length of the formula f as a string over $Prop \cup \{\wedge, \vee, \neg, \Rightarrow, \bigcirc, \Box, \Diamond, \mathcal{U}, \mathcal{W}\}$, then $deg(f) < 2|f|$)

2.1.2 Semantics.

A model S in $\mathcal{FL}$ is a pair (s, v) where s is a denumerable sequence $s_0, s_1, s_2, \ldots$ of states and v is a map

$$v: s \rightarrow 2^{Prop}.$$

($v(s_j)$ is the set of propositions which are true in state s_j).

An interpretation is a pair $\langle \mathcal{S}, s_j \rangle$ where s_j is a state of S.

The truth of a formula f in an interpretation $\langle \mathcal{S}, s_j \rangle$ is denoted by

$$\langle \mathcal{S}, s_j \rangle \models f$$

For any formula f and interpretation $\langle \mathcal{S}, s_j \rangle$, either $\langle \mathcal{S}, s_j \rangle \models f$ or $\langle \mathcal{S}, s_j \rangle \not\models f$.

$\models$ is defined as follows

Definition 2 *For $f, g \in \mathcal{FL}$, $p \in Prop$*

$$
\begin{array}{lcll}
\langle \mathcal{S}, s_j \rangle & \models & \textbf{true}. & \\
\langle \mathcal{S}, s_j \rangle & \not\models & \textbf{false}. & \\
\langle \mathcal{S}, s_j \rangle & \models & p & \text{iff} \quad p \in v(s_j) \\
\langle \mathcal{S}, s_j \rangle & \models & \neg f & \text{iff} \quad \langle \mathcal{S}, s_j \rangle \not\models f \\
\langle \mathcal{S}, s_j \rangle & \models & f \wedge g & \text{iff} \quad \langle \mathcal{S}, s_j \rangle \models f \text{ and } \langle \mathcal{S}, s_j \rangle \models g \\
\langle \mathcal{S}, s_j \rangle & \models & \bigcirc f & \text{iff} \quad \langle \mathcal{S}, s_{j+1} \rangle \models f \\
\langle \mathcal{S}, s_j \rangle & \models & f \,\mathcal{U} g & \text{iff there is a } k, k \geq j, \text{ with } \langle \mathcal{S}, s_k \rangle \models g \text{ and} \\
& & & \text{for all } i \text{ with } j \leq i < k,\ \langle \mathcal{S}, s_i \rangle \models f
\end{array}
$$

If $\langle \mathcal{S}, s_j \rangle \models f$ for all interpretations $\langle \mathcal{S}, s_j \rangle$ then f is said to be *valid* and we write $\models f$.

Definition 3 *Define the relation* $\equiv$ (truth functional equivalence) *on* $\mathcal{FL}$ *by* $f \equiv g$ *iff f and g are satisfied in exactly the same interpretations. i.e.*$\langle \mathcal{S}, s_j \rangle \models f$ *iff* $\langle \mathcal{S}, s_j \rangle \models g$

(Note that $f \equiv g$ iff $\models f \Leftrightarrow g$)

The semantics of the other operators is defined in terms of the above primitive operators. For $f, g \in \mathcal{FL}$

$$
\begin{array}{rcl}
f \vee g & \equiv & \neg(\neg f \wedge \neg g) \\
f \Rightarrow g & \equiv & \neg f \vee g \\
f \Leftrightarrow g & \equiv & (f \Rightarrow g) \wedge (g \Rightarrow f) \\
\Diamond f & \equiv & \textbf{true}\, \mathcal{U} f \\
\Box f & \equiv & \neg \Diamond \neg f \\
f \,\mathcal{W} g & \equiv & f \,\mathcal{U} g \vee \Box f
\end{array}
$$

2.2 Semantic Properties.

In this section we present a collection of lemmas for use in the subsequent discussion of the decision procedures.

Lemma 1 *If f is a formula in $\mathcal{FL}$ involving the subformula g, and h is a formula such that $g \equiv h$, then*

$$f \equiv f[h \backslash g].$$

where $f[h \backslash g]$ denotes the formula given by replacing occurrences of g in f by h.

Proof

We need to show that if $f, g, h \in \mathcal{FL}$ and $f \equiv g$, then

$$
\begin{array}{c}
\neg f \equiv \neg g \\
f \wedge h \equiv g \wedge h \\
\bigcirc f \equiv \bigcirc g \\
f \,\mathcal{U} h \equiv g \,\mathcal{U} h \\
h \,\mathcal{U} f \equiv h \,\mathcal{U} g
\end{array}
$$

Each of the above is an immediate consequence of the definitions. ∎

Lemma 2
For all $f, g, h \in \mathcal{FL}$.

$$\begin{array}{rrcl}
i) & \neg\textbf{false} & \equiv & \textbf{true}. \\
ii) & \neg\textbf{true} & \equiv & \textbf{false}. \\
iii) & \neg\neg f & \equiv & f. \\
iv) & f \wedge f & \equiv & f \\
v) & f \wedge (g \wedge h) & \equiv & (f \wedge g) \wedge h. \\
vi) & f \wedge g & \equiv & g \wedge f. \\
vii) & \neg\bigcirc f & \equiv & \bigcirc\neg f. \\
viii) & \bigcirc(f \wedge g) & \equiv & \bigcirc f \wedge \bigcirc g. \\
ix) & (f\,\mathcal{U} g) & \equiv & g \vee (f \wedge \bigcirc(f\,\mathcal{U} g)). \\
x) & \neg(f\,\mathcal{U} g) & \equiv & \neg g\,\mathcal{W}(\neg f \wedge \neg g). \\
xi) & (f\,\mathcal{W} g) & \equiv & g \vee (f \wedge \bigcirc(f\,\mathcal{W} g)). \\
xii) & \neg(f\,\mathcal{W} g) & \equiv & \neg g\,\mathcal{U}(\neg f \wedge \neg g). \\
xiii) & \Box f & \equiv & f \wedge \bigcirc\Box f. \\
xiv) & \neg\Box f & \equiv & \Diamond\neg f. \\
xv) & \Diamond f & \equiv & f \vee \bigcirc\Diamond f. \\
xvi) & \neg\Diamond f & \equiv & \Box\neg f.
\end{array}$$

Proof

i)-viii) are immediate consequences of the definitions.

Proof of ix)

a) $(\Rightarrow)$
Suppose $\langle \mathcal{S}, s_j\rangle \models (f\,\mathcal{U} g)$, then

$$\exists k \geq j \text{with } \langle \mathcal{S}, s_k\rangle \models g, \text{and } \forall i \text{with } j \leq i < k, \langle \mathcal{S}, s_i\rangle \models f.$$

This gives two possibilities, either

$$\text{i) } k = j \quad \text{or ii) } k \geq j+1.$$

In case i) we have $\langle \mathcal{S}, s_j\rangle \models g$. In case ii) we have $j < k$ and so $\langle \mathcal{S}, s_j\rangle \models f$ and there is a $k \geq j+1$ with $\langle \mathcal{S}, s_k\rangle \models g$, and for all i with $j+1 \leq i < k$, $\langle \mathcal{S}, s_i\rangle \models f$. i.e. $\langle \mathcal{S}, s_{j+1}\rangle \models f\,\mathcal{U} g$ and so $\langle \mathcal{S}, s_j\rangle \models \bigcirc(f\,\mathcal{U} g)$. Therefore $\langle \mathcal{S}, s_j\rangle \models (f \wedge \bigcirc(f\,\mathcal{U} g))$. i) or ii) gives $\langle \mathcal{S}, s_j\rangle \models g \vee (f \wedge \bigcirc(f\,\mathcal{U} g))$.

b) $(\Leftarrow)$
If $\langle \mathcal{S}, s_j\rangle \models g \vee (f \wedge (\bigcirc(f\,\mathcal{U} g))$, then either i) $\langle \mathcal{S}, s_j\rangle \models g$, or ii) $\langle \mathcal{S}, s_j\rangle \models f$ and $\langle \mathcal{S}, s_{j+1}\rangle \models f\,\mathcal{U} g$. We need to show that, in either case there is a $k \geq j$ with $\langle \mathcal{S}, s_k\rangle \models g$, and for all i with $j \leq i < k$, $\langle \mathcal{S}, s_i\rangle \models f$. In case i), we simply take $k = j$. In case ii), since $\langle \mathcal{S}, s_{j+1}\rangle \models f\,\mathcal{U} g$ we know that there is a $k \geq j$ with $\langle \mathcal{S}, s_k\rangle \models g$, and for all i with $j \leq i < k$, $\langle \mathcal{S}, s_i\rangle \models f$. For such a k we have $\langle \mathcal{S}, s_j\rangle \models f$ and for all i with $j+1 \leq i < k$, $\langle \mathcal{S}, s_i\rangle \models f$. Therefore for all i with $j \leq i < k$, $\langle \mathcal{S}, s_i\rangle \models$ f and hence $\langle \mathcal{S}, s_j\rangle \models f\,\mathcal{U} g$.

Proof of x)

a) ($\Rightarrow$)
Suppose $\langle \mathcal{S}, s_j \rangle \models \neg(f\, \mathcal{U} g)$, then $\langle \mathcal{S}, s_j \rangle \not\models (f\, \mathcal{U} g)$, so that, for all $k \geq j$, $\langle \mathcal{S}, s_k \rangle \models \neg g$ or there is an i with $j \leq i < k$ and $\langle \mathcal{S}, s_i \rangle \models \neg f$. So, either i) for all $k \geq j$, $\langle \mathcal{S}, s_k \rangle \models \neg g$, in which case $\langle \mathcal{S}, s_k \rangle \models \Box\neg g$. or ii) there is a $k_1 \geq j$ with $\langle \mathcal{S}, s_{k_1} \rangle \models g$ and, for all i, $j \leq i < k_1$, $\langle \mathcal{S}, s_i \rangle \models \neg g$. Now, since $\langle \mathcal{S}, s_{k_1} \rangle \not\models \neg g$ it must be the case that there is a k_2, $j \leq k_2 < k_1$ with $\langle \mathcal{S}, s_{k_2} \rangle \models \neg f$. Since $k_2 < k_1$, $\langle \mathcal{S}, s_{k_2} \rangle \models \neg g$ and therefore $\langle \mathcal{S}, s_{k_2} \rangle \models \neg f \wedge \neg g$ and for all i, $j \leq i < k_2$ $\langle \mathcal{S}, s_i \rangle \models \neg g$ Hence $\langle \mathcal{S}, s_j \rangle \models \neg g\, \mathcal{U}(\neg f \wedge \neg g)$. i) or ii) gives $\langle \mathcal{S}, s_j \rangle \models (\neg g\, \mathcal{U}(\neg f \wedge \neg g)) \vee \Box\neg g$.

b) ($\Leftarrow$)
Suppose

$$\langle \mathcal{S}, s_j \rangle \models (\neg g\, \mathcal{U}(\neg f \wedge \neg g)) \vee \Box\neg g,$$

then either i) $\langle \mathcal{S}, s_j \rangle \models (\neg g\, \mathcal{U}(\neg f \wedge \neg g))$, or ii) $\langle \mathcal{S}, s_j \rangle \models \Box\neg g$, In case i) there is a $k_1 \geq j$ with $\langle \mathcal{S}, s_{k_1} \rangle \models \neg f \wedge \neg g$ and, for all i with $j \leq i < k_1$, $\langle \mathcal{S}, s_i \rangle \models \neg g)$. So for all k with $j \leq k < k_1$, $\langle \mathcal{S}, s_k \rangle \models \neg g$ and for all $k \geq k_1$, there is an i with $j \leq i < k$, $\langle \mathcal{S}, s_i \rangle \models \neg f$. (we can take $i = k_1$) Therefore, for all k with $k \geq j$, $\langle \mathcal{S}, s_k \rangle \models \neg g$ or there is an i with $j \leq i < k$ and $\langle \mathcal{S}, s_i \rangle \models \neg f$, and so $\langle \mathcal{S}, s_j \rangle \models \neg(f\, \mathcal{U} g)$. In case ii) for all k with $k \geq j$, $\langle \mathcal{S}, s_k \rangle \models \neg g$ therefore for all k with $k \geq j$, $\langle \mathcal{S}, s_k \rangle \models \neg g$ or there is an i with $j \leq i < k$ and $\langle \mathcal{S}, s_i \rangle \models \neg f$, and so $\langle \mathcal{S}, s_j \rangle \models \neg(f\, \mathcal{U} g)$.

∎

2.2.1 Normalisation.

Using the above results we see that any formula in $\mathcal{FL}$ is truth-functionally equivalent to a formula in which $\neg$ is applied only to atomic propositions. We assume in the following that all formulae are in this normalised form.

2.3 Alpha and Beta formulae.

We can now classify all normalised formulae in $\mathcal{FL}$ into four types :-

i) Literals, i.e. propositions and their negations.

ii) Next-time formulae, i.e. formulae whose leading connective is $\bigcirc$.

iii) α-formulae, i.e. formulae which are truth functionally equivalent to conjunctions.

iv) β-formulae, i.e. formulae which are truth functionally equivalent to disjunctions.

We make this classification explicit in the following tables

2.3.1 α and β tables

α	α_1	α_2	β	β_1	β_2
$p \wedge q$	p	q	$p \vee q$	p	q
$\Box p$	p	$\bigcirc \Box p$	$\Diamond p$	p	$\bigcirc \Diamond p$
			$p\, \mathcal{U} q$	q	$p \wedge \bigcirc(p\, \mathcal{U} q)$
			$p\, \mathcal{W} q$	q	$p \wedge \bigcirc(p\, \mathcal{W} q)$

If f is an α (or β) formula then the formulae formed according to the rules α_1 and α_2 (or β_1 and β_2) are called the α-components (β-components) of f.

2.3.2 Alpha Equivalence.

We define a function

$$\alpha^{*}: \mathcal{F}\mathcal{L} \rightarrow 2^{\mathcal{F}\mathcal{L}}$$

by

$$\begin{array}{rl} \alpha^{*}(f) = \{f\} & \text{if } f \text{ is a literal, a } \beta \text{ formula or a nexttime formula} \\ \alpha^{*}(f) = \alpha^{*}(f_1) \cup \alpha^{*}(f_2) & \text{if } f \text{ is an } \alpha \text{ formula with components } f_1, f_2 \end{array}$$

Lemma 3 *$\langle \mathcal{S}, s_j \rangle \models f$ iff $\langle \mathcal{S}, s_j \rangle \models \alpha^{*}(f)$.*
(where $\langle \mathcal{S}, s_j \rangle \models Q$, for Q a subset of $\mathcal{F}\mathcal{L}$ iff for all $q \in Q$, $\langle \mathcal{S}, s_j \rangle \models q$).

Proof

A simple induction on the degree of f provides the proof. ∎

2.3.3 Eventualities.

In any interpretation in which either of the formulae $f\, \mathcal{U} g$ or $\Diamond g$ is satisfied there must be a future state in which g is satisfied. To restate this more precisely, we have

If $\langle \mathcal{S}, s_j \rangle \models f\, \mathcal{U} g$, then there is a k, with $k \geq j$ and $\langle \mathcal{S}, s_k \rangle \models g$.

If $\langle \mathcal{S}, s_j \rangle \models \Diamond g$, then there is a k, with $k \geq j$ and $\langle \mathcal{S}, s_k \rangle \models g$.

Both of the above formulae are β-formulae with β_1 component g. Denote either formula by h and its β components by h_1 and h_2, so that $h_1 = g$.

The formula h is satisfied in an interpretation if and only if either of h_1 or h_2 is satisfied, but can not be satisfied in an interpretation consisting of an infinite sequence of states in which h_2 is satisfied and h_1 is not satisfied ; both formulae imply that eventually there is a state in which h_1 is satisfied.

We say that $\Diamond g$ and $f\, \mathcal{U} g$ have an associated eventuality g (or eventuality set $\alpha^{*}(g)$).

2.4 Properties of subsets of $\mathcal{F}\mathcal{L}$.

We will be concerned greatly in the subsequent sections with objects labelled with various subsets of $\mathcal{F}\mathcal{L}$. We now consider certain useful properties of such subsets, which we will refer to as L-sets.

2.4.1 Propriety.

An L-set FF is said to be *proper* iff it satisfies

i) If p is a proposition and $\neg p \in FF$, then $p \notin FF$.

ii) **false** $\notin FF$.

Essentially an L-set is proper if it contains no immediate contradictions.

2.4.2 Downward closure.

Definition 4 *An L-set FF is said to be* downward closed *iff it satisfies*

DCi) FF contains no α*-formulae.*

DCii) If $f \in FF$ is a β*-formula, with* β*-components f_1 and f_2 then*

$$\alpha^*(f_1) \subseteq FF \text{ or } \alpha^*(f_2) \subseteq FF$$

For example, if $p, q \in Prop$, the sets $\{\Diamond p \vee \Diamond q, \Diamond q, q\}$ and $\{p\,\mathcal{U} q, p, \bigcirc(p\,\mathcal{U} q)\}$ are downward closed, but $\{\Diamond p \vee \Diamond q, \Diamond q, p\}$ is not.

Given an L-set FF containing no α-formulae there are many downward closed L-sets containing FF. We now introduce the idea of a downward closure of an L-set FF, a downward closed L-set containing FF which is in some sense minimal.

Definition 5 *If T is an L-set with subset T_1 then $f \in T$ is* necessary *for T_1 if one of the following conditions holds:-*

i) $f \in T_1$

ii) There is a β*-formula g which is necessary for T_1 (with components g_1 and g_2) and either*

a) $f \in \alpha^*(g_1) \cap \alpha^*(g_2)$.

or b) $f \in \alpha^*(g_1) \subseteq T$ and $\alpha^*(g_2) \not\subseteq T$

or c) $f \in \alpha^*(g_2) \subseteq T$ and $\alpha^*(g_1) \not\subseteq T$

or iii) There is a β-formula g which is necessary for T_1 with eventuality g_1 and

$$f \in \alpha^*(g_1) \subseteq T.$$

Definition 6 *An L-set FF_1 is a* downward closure *of an L-set FF iff*

i) $\alpha^*(FF) \subseteq FF_1$.

ii) FF_1 is downward closed.

iii) All formulae in FF_1 are necessary for $\alpha^*(FF)$

e.g. If $FF = \{(\neg p \vee q), r\}$ where $p, q, r \in Prop$ then the downward closures of FF are $\{(\neg p \vee q), \neg p, r\}$ and $\{(\neg p \vee q), q, r\}$

Denote by $DC(FF)$ the set of all downward closures of the L-set FF, and by $PDC(FF)$ the subset of $DC(FF)$ consisting of proper L-sets.

The importance of the downward closure construction is highlighted by the following result:-

Theorem 1 *A formula f in PC is satisfiable iff $PDC(\{f\})$ is non-empty.*

PC denotes the sub-language of $\mathcal{FL}$ consisting of formulae containing no temporal operators. This theorem is a consequence of a stronger result which we prove later.

For example if

$$f = \neg(((p \Rightarrow q) \wedge (q \Rightarrow r)) \Rightarrow (p \Rightarrow r))$$

then

$$f \equiv (\neg p \vee q) \wedge (\neg q \vee r) \wedge (p \wedge \neg r),$$

and

$$\begin{aligned} DC(\{f\}) = \; & \{\{(\neg p \vee q), (\neg q \vee r), (p \wedge \neg r), \neg p, \neg q, p, \neg r\}, \\ & \{(\neg p \vee q), (\neg q \vee r), (p \wedge \neg r), \neg p, r, p, \neg r\}, \\ & \{(\neg p \vee q), (\neg q \vee r), (p \wedge \neg r), q, \neg q, p, \neg r\}, \\ & \{(\neg p \vee q), (\neg q \vee r), (p \wedge \neg r), q, r, p, \neg r\}\} \end{aligned}$$

and so $PDC(\{f\}) = \emptyset$.

Therefore f is not satisfiable, and so $\neg f$ is valid.

2.4.3 Construction of the downward closure.

Given an L-set FF the following algorithm will generate the set $DC(FF)$.

i) If FF contains α-formulae then replace FF with $\alpha^*(FF)$.

ii) Set $D = \{FF\}$ and all β-formulae as unmarked.

iii) If $T \in D$ and $f \in T$ is an unmarked β-formula with components f_1 and f_2 then replace T in D by the sets $T_1 = T \cup \alpha^*(f_1)$ and $T_2 = T \cup \alpha^*(f_2)$ in which the formula f is marked.

iv) Repeat iii) until each set in D consists entirely of marked β-formulae, nexttime formulae and literals.

v) Delete any sets from D which contain formulae which are not necessary for $\alpha^*(FF)$.
On termination we have $D = DC(FF)$.
(This process must terminate since all new β-formulae introduced at stage iii) have lower degree than f).

For example if $FF = \{p\,\mathcal{U}q, \Diamond q, \Diamond p\}$ then stages i) - iv) of the above algorithm generates the set

$$\begin{array}{lll} D = & \{\{p\,\mathcal{U}q, \Diamond q, \Diamond p, q, p\}, & \\ & \{p\,\mathcal{U}q, \Diamond q, \Diamond p, q, \bigcirc\Diamond p\}, & \\ & \{p\,\mathcal{U}q, \Diamond q, \Diamond p, q, \bigcirc\Diamond q, p\}, & ** \\ & \{p\,\mathcal{U}q, \Diamond q, \Diamond p, q, \bigcirc\Diamond q, \bigcirc\Diamond p\}, & ** \\ & \{p\,\mathcal{U}q, \Diamond q, \Diamond p, p, \bigcirc(p\,\mathcal{U}q), q\}, & ** \\ & \{p\,\mathcal{U}q, \Diamond q, \Diamond p, p, \bigcirc(p\,\mathcal{U}q), q, \bigcirc\Diamond p\}, & ** \\ & \{p\,\mathcal{U}q, \Diamond q, \Diamond p, p, \bigcirc(p\,\mathcal{U}q), \bigcirc\Diamond q\}, & \\ & \{p\,\mathcal{U}q, \Diamond q, \Diamond p, p, \bigcirc(p\,\mathcal{U}q), \bigcirc\Diamond q, \bigcirc\Diamond p\} & **\} \end{array}$$

but stage v) deletes the starred sets to leave

$$\begin{array}{ll} DC(FF) = & \{\{p\,\mathcal{U}q, \Diamond q, \Diamond p, q, p\}, \\ & \{p\,\mathcal{U}q, \Diamond q, \Diamond p, q, \bigcirc\Diamond p\}, \\ & \{p\,\mathcal{U}q, \Diamond q, \Diamond p, p, \bigcirc(p\,\mathcal{U}q), \bigcirc\Diamond q\}\} \end{array}$$

2.4.4 The Fischer-Ladner closure.

Definition 7 *For any $u \in \mathcal{F}L$ we define the* Fischer-Ladner closure *of u [FL79] to be the L-set $FL(u)$ satisfying*

FLi) If $\neg g \in FL(u)$ then $g \in FL(u)$.

FLii) If $f \in FL(u)$ is an α-formula with α-components f_1 and f_2, then $f_1, f_2 \in FL(u)$.

FLiii) If $f \in FL(u)$ is an β-formula with β-components f_1 and f_2, then $f_1, f_2 \in FL(u)$.

FLiv) If $\bigcirc g \in FL(u)$ then $g \in FL(u)$.

For example if $u = p \vee (r\,\mathcal{U}s)$ then

$$FL(u) = \{(p \vee (r\,\mathcal{U}s)), p, (r\,\mathcal{U}s), s, r \wedge \bigcirc(r\,\mathcal{U}s), r, \bigcirc(r\,\mathcal{U}s)\}$$

For a given u the set $FL(u)$ is not necessarily downward closed since it may contain α-formulae, but it does satisfy a stronger condition than DCii), namely that

If $f \in FL(u)$ is a β-formula, with β-components f_1 and f_2 then

$$\alpha^*(f_1) \subseteq FL(u) \text{ and } \alpha^*(f_2) \subseteq FL(u)$$

From the above we can immediately deduce the following lemma

Lemma 4 *If $F = FL(u)$, for any $\mathcal{FL}$ formula u, and C is any downward closed L-set, then $C \cap F$ is also downward closed.*

An easy induction on the structure of u shows that the size of the set $FL(u)$, $|FL(u)|$, is bounded above, and that, for all formulae u

$$|FL(u)| \leq deg(u) < 2|u|.$$

2.5 A decision procedure for $\mathcal{FL}$.

The decision procedure presented here is based on the semantic tableau method [Smu68, RU71, HC72], and is similar to that presented in [Wol83], although with several important differences.

As with all semantic tableau based methods, a formula f is proved to be valid by attempting to construct a model for the formula $\neg f$. f is shown to be valid if it is not possible to construct such a model, since for any interpretation $\langle \mathcal{S}, s_j \rangle$ and formula f we must have either

$$\langle \mathcal{S}, s_j \rangle \models f \text{ or } \langle \mathcal{S}, s_j \rangle \models \neg f.$$

The decision procedure provides an algorithm for the construction of a model for a given formula.

The main idea behind the construction is that any formula can be decomposed to give a collection of sets containing formulae that either apply to the current time alone, or are nexttime formulae (i.e have $\bigcirc$ as their leading connective).This decomposition separates the requirements of a formula f into present requirements and requirements for the rest of the sequence (future commitments).

Given a formula f for which we require a model, we build a graph in which each node represents a state and is labelled by a set of formulae which are true in that state. A model for f can then be constructed by by locating a node in the graph containing f (or a set of equivalent formulae), and tracing a path through the graph . This path is not quite a model for f since the states are labelled with formulae other than propositions . We call such a path a Hintikka structure and show that a formula $f \in \mathcal{FL}$ has a model if and only if it has a Hintikka structure.

2.5.1 Hintikka Structures.

All decision procedures based on semantic tableaux construct not a model for the given formula but a structure closely resembling a model called a Hintikka structure .

Definition 8 *A* Hintikka structure *(over the language $\mathcal{FL}$) is a pair H=(s, L), where*

$$s = s_0, s_1, s_2 \ldots$$

is an infinite sequence of states and

$$L\colon s \to 2^{\mathcal{FL}}$$

is a labelling of states with formulae, which satisfies, for all $i \geq 0$

Hi) *$L(s_i)$ is proper.*

Hii) *$L(s_i)$ is downward closed.*

Hiii) *If $\bigcirc f \in L(s_i)$ then $\alpha^*(f) \subseteq L(s_{i+1})$.*

Hiv) *If $f \in L(s_i)$ with eventuality e, then there is a $j \leq i$ such that $\{f\} \cup \alpha^*(e) \subseteq L(s_j)$.*

H is said to be a Hintikka structure for the formula f if $\alpha^*(f) \subseteq L(s_0)$.

A Hintikka structure for f is said to be *complete* iff for all i,

$$L(s_i) = \{u \mid u \in FL(f) \text{ and } \langle \mathcal{S}, s_i \rangle \models u\}$$

(where $FL(f)$ denotes the Fischer-Ladner closure of the formula f)

(N.B. Any Hintikka structure for f can be made into a complete Hintikka structure in the obvious way, by adding to $L(s_i)$ all formulae $g \in FL(f)$ such that $\langle \mathcal{S}, s_i \rangle \models g$.)

Theorem 2 *Hintikka's Lemma for $\mathcal{FL}$.*

A formula of $\mathcal{FL}$ is satisfiable iff there exists a (complete) Hintikka structure for f.

Proof

a) (only if)
Any model is also a Hintikka structure.

b) (if)
Suppose there exists a Hintikka structure $H = (s, L)$ for f, we wish to construct a model for f.

Define a function

$$v\colon s \to 2^P,$$

an assignment of propositions for the desired model, by

$$p \in v(s_i) \text{ iff } p \in L(s_i).$$

that is we pick out the propositions from L(s_i).

We now show that for any formula $g \in L(s_i)$, $\langle \mathcal{S}, s_i \rangle \models g$ and therefore $\langle \mathcal{S}, s_0 \rangle \models f$, in this model.

Suppose there exists a state s_i and formula f, with $f \in L(s_i)$ and $\langle \mathcal{S}, s_i \rangle \not\models f$. Let s_i and f be such a pair, with the additional property that f has minimal degree, so that, for all $j \geq 0$, if $g \in L(s_j)$ has lower degree than f, then $\langle \mathcal{S}, s_j \rangle \models g$.

Now f cannot be atomic (by the construction of v), therefore f is either a β-formula or a nexttime formula. In fact it cannot be a nexttime formula, since if $f = \bigcirc g$, then $\langle \mathcal{S}, s_i \rangle \not\models f$ iff $\langle \mathcal{S}, s_{i+1} \rangle \not\models g$, and g has lower degree than f, which contradicts the minimality of $deg(f)$. f is therefore a β-formula and we denote by f_1, f_2 the β-components of f. Since, for all β-formulae, $deg(f_1) < deg(f)$, we must have $\alpha^*(f_2) \subseteq L(s_i)$, and a formula $g \in \alpha^*(f_2)$ such that $\langle \mathcal{S}, s_i \rangle \not\models g$, and therefore $deg(g) >= deg(f)$.From the β-table we have the following possibilities

$$\begin{array}{lcll} f & = & u \vee v & \text{and} \quad g \in \alpha^*(v), \\ f & = & \Diamond u & \text{and} \quad g = \bigcirc\Diamond u, \\ f & = & u\,\mathcal{U} v & \text{and} \quad g \in \alpha^*(u) \text{ or } g = \bigcirc(u\,\mathcal{U} v), \quad \text{or} \\ f & = & u\,\mathcal{W} v & \text{and} \quad g \in \alpha^*(u) \text{ or } g = \bigcirc(u\,\mathcal{W} v), \end{array}$$

and so the only candidate for a g with $deg(g) \geq deg(f)$ is the formula $\bigcirc f$. By induction we see that this implies that $\langle \mathcal{S}, s_k \rangle \not\models f$ for all $k \geq i$. Now, if f is a formula with eventuality e, condition Hiv) ensures that $\alpha^*(e) \subseteq L(s_j)$ for some $j \geq i$ and since the degree of every formula in $\alpha^*(e)$ will be less than that of f we have $\langle \mathcal{S}, s_j \rangle \models e$, and hence $\langle \mathcal{S}, s_j \rangle \models f$, which gives a contradiction. Otherwise f must be a formula of the form u $\mathcal{W}$v and we have shown that, for all $k \geq i, \alpha^*(f_2) \subseteq L(s_k)$. But

$$\alpha^*(f_2) = \alpha^*(u \wedge \bigcirc(u\,\mathcal{W} v)) = \alpha^*(u) \cup \{\bigcirc(u\,\mathcal{W} v)\},$$

and we know that all formulae in $\alpha^*(u)$ have lower degree than deg(f), therefore $\langle \mathcal{S}, s_k \rangle \models u$ for all $k \geq i$, and so $\langle \mathcal{S}, s_i \rangle \models (u\,\mathcal{W} v)$. This gives the desired contradiction.

This concludes the proof of the theorem. ∎

2.5.2 H-Graphs.

To obtain the required Hintikka structure we construct a finite graph and then choose an infinite path within the graph to give the Hintikka structure. We now define the properties of the graphs to be used.

Definition 9 *A* pre-H-graph *is a triple G = (S, L, R) where*

> *S is a finite set of states,*
>
> *$L: S \rightarrow 2^{\mathcal{FL}}$ is a labelling of the states,*
>
> *and $R \subseteq S \times S$ is a relation on S.*

satisfying

HGi) For all s, L(s) is proper,

HGii) For all s, L(s) is downward closed.

HGiii) For all s, if $\bigcirc f \in L(s)$ *then, for all s1 with* $(s,s1) \in R$, $\alpha^*(f) \subseteq L(s1)$.

Definition 10 *If s is a state in the pre-H-graph* $G = (S,L,R)$ *and* $f \in L(s)$ *is a formula with eventuality ev then ev is said to be* resolvable *in G if there is a state* $s1$ *with* $(s,s1) \in R^*$ *and* $\{f\} \cup \alpha^*(ev) \subseteq L(s1)$ *. (Where* R^* *denotes the reflexive, transitive closure of R.)*

Definition 11 *A pre-H-graph G = (S, L, R) is said to be an* H-graph *if it satisfies the additional properties*

HGiv) R is total ; i.e. for all $s \in S$ *there is an* $s1 \in S$ *with* $(s,s1) \in R$

HGv) If $f \in L(s)$ *has an eventuality ev, then ev is resolvable in G .*

Definition 12 *A* path *in an H-graph G is a finite or infinite sequence*

$$s_0,\ s_1,\ s_2,\ \ldots$$

where, for all i, $(s_i, s_{i+1}) \in R$*.*
A path from s to t, where $s,t \in S$*, is a path*

$$s_0, s_1 \ldots s_k$$

with $s_0 = s$ *and* $s_k = t$*.*

Lemma 5 *f has a Hintikka structure if and only if there exists an H-graph containing a state* s_0 *with* $\alpha^*(f) \subseteq L(s_0)$*.*

Proof

a) (only if) This is fairly straight forward. Simply choose an infinite path, starting at s_0, on which all eventualities are resolved .
b) (if) Let H = (s, L) be a Hintikka structure for f, and let F be the Fischer-Ladner closure of f . We define an equivalence relation RF on the sequence s by

$$(s_i, s_j) \in RF \text{ iff } L(s_i) \cap F = L(s_j) \cap F.$$

Let $S_1 = s/RF$, and let $[s_i]$ denote the equivalence class containing the state s_i. We now define a map $L_1\colon S_1 \to 2^{\mathcal{F}\mathcal{L}}$ by

$$L_1([s_i]) = L(s_i) \cap F.$$

This map is clearly well-defined. Finally we define a relation R_1 on S_1 by

$$R_1 = \{([s_i],[s_{i+1}]) \mid i \geq 0\}$$

and claim that $G = (S_1, L_1, R_1)$ is the required H-graph.We need to check that G satisfies properties HGi)-v).

HGi) is an immediate consequence of Hi) and HGii) follows from Hii) and Lemma 2.4.4.1. The definition of R_1 and Hiii) imply HGiii) and HGiv), and HGv) follows from Hiv) and the observation that if $i > j$ then $([s_i],[s_j]) \in R_1^*$. This completes the proof. ∎

2.5.3 The decision procedure.

We describe an algorithm for the construction of an H-graph, $G = (S, L, R)$ for a given formula u.

i) (Initialisation)
Construct $PDC(\alpha^*(u))$, then for each $c \in PDC(\alpha^*(u))$, add a new state s to S with $L(s)$ defined to be c. Set $R = \emptyset$.
ii) (Graph Construction)
For each $s \in S$, define $nexts(s)$ by

$$nexts(s) = \bigcup\{\alpha^*(f) | \bigcirc f \in L(s)\},$$

and construct $PDC(nexts(s))$. For each $c \in PDC(nexts(s))$ add a state $s1$ with $L(s1) = c$ (unless such a state already exists) and add $(s, s1)$ to R.
iii) (Graph contraction)
Delete all states s satisfying

a) s has no successors. i.e. there is no $s1$ with $(s, s1) \in R$.

or

b) $L(s)$ contains an eventuality ev which is not resolvable in G.

iv) Repeat iii) until no further deletions are possible.

The algorithm terminates, since stages i) and ii) construct a graph whose nodes are in one to one correspondence with a subset of $2^{FL(u)}$, and is hence finite, and stage iii) merely contracts that graph. The structure produced by the algorithm is an H-graph, since the structure generated by stages i) and ii) clearly satisfies HGi)-iii), and is therefore a pre-H-graph and stage iii) ensures that HGiv) and HGv) are satisfied. We further claim that for any satisfiable $\mathcal{FL}$ formula u, the algorithm will construct an H-graph for u, that is that the algorithm is consistent. We justify this claim in the following section.

2.6 Consistency of the algorithm

We need to show that if u is satisfiable then the algorithm is consistent i.e.

Theorem 3 *An $\mathcal{FL}$ formula is satisfiable iff the algorithm constructs an H-graph for f.*

Before proving this result we state and prove a lemma.

Lemma 6 *Suppose T is an L-set, T_1 a subset of T and $E = ev(T_1)$ denotes the union of the eventuality sets of all formulae in T_1. If $TE = T \cup E$, then $PDC(TE) \subseteq PDC(T)$.*

Proof

Let $S \in PDC(TE)$, then S is downward closed and $T \cup E \subseteq S$ and therefore $T \subseteq S$. Now all formulae in S are necessary for $T \cup E$, and since all formulae in E are necessary for T, all formulae in S must be necessary for T. This completes the proof of the lemma, we now prove the theorem. ∎

Proof

(**Theorem 3**) If f is satisfiable then we can find a complete Hintikka structure (m,V) for f. Denote by $G_0 = (S_0, L_0, R_0)$ the pre-H-graph generated by stages i) and ii) of the algorithm. We construct a map

$$\theta : m \to S_0$$

such that if $S_1 = \theta(m)$ and L_1 and R_1 denote the restrictions of L_0 and R_0 to S_1, then $G_1 = (S_1, L_1, R_1)$ is an H-graph for f.

Let $F(0) \subseteq \alpha^*(f)$ be the set of all formulae in $\alpha^*(f)$ whose eventualities are satisfied at m_0, and $TE = (\alpha^*(f) \cup ev(F(0)))$. By the previous lemma there is $c \in PDC(\alpha^*(f))$ with

$$TE \subseteq c \subseteq V(m_0)$$

and we define $\theta(m_0)$ to be the state s_0 with $L_0(s_0) = c$. Such a state must exist since otherwise c and hence $V(m_0)$ would be improper.

We now proceed inductively. Suppose we have defined $\theta(m_i)$ for $0 \le i \le k$. If $\theta(m_k) = s_k$ then $nexts(s_k) \subseteq nexts(m_k)$ and $nexts(m_k) \subseteq V(m_{k+1})$; therefore if F(k+1) denotes those formulae in $nexts(s_k)$ whose eventualities are satisfied in m_{k+1}, we can find a $c \in PDC(nexts(s_k))$ with $nexts(s_k) \cup e(F(k+1)) \subseteq c \subseteq V(m_{k+1})$. Define s_{k+1} to be the state with $L_0(s_{k+1}) = c$.

θ is therefore defined, and for each i, $(\theta(m_i), \theta(m_{i+1})) \in R$.

Now if $S_1 = \theta(m)$, and L_1, R_1 are defined as above, we claim that (S_1, L_1, R_1) is an H-graph for f and therefore none of the states of S_1 will be eliminated at stage iii) of the algorithm. By construction every $s \in S_1$ has a successor. If $f \in L_1(s)$ has eventuality g and $s = (m_i)$ then there is a $k \ge i$ with $\{f\} \cup \alpha^*(g) \subseteq V(m_k)$ and so, if $s_1 = \theta(m_k)$ then $(s,\ s_1) \in R^*$ and so f is resolvable. This completes the proof. ∎

We conclude this chapter with two worked examples, one of the use of the decision procedure to prove a formula valid, and the second of the construction of a model for a satisfiable formula.

2.7 Example 1.

Let ϕ be the formula defined by

$$\phi = (\Diamond p \wedge \Box(p \Rightarrow \bigcirc p)) \Rightarrow \Diamond \Box p.$$

We use the decision procedure to show that ϕ is valid. After normalisation we see that

$$\begin{aligned} \neg\phi \quad &\Leftrightarrow \quad \Diamond p \wedge \Box(\neg p \vee \bigcirc p) \wedge \Box\Diamond(\neg p) \\ &\Leftrightarrow \quad \Diamond p \wedge \Diamond\neg p \wedge (\neg p \vee \bigcirc p) \wedge \bigcirc\Box(\neg p \vee \bigcirc p) \wedge \bigcirc\Box\Diamond\neg p \end{aligned}$$

Therefore

$$\alpha^*(\neg\phi) = \{\Diamond p, \Diamond\neg p, (\neg p + \bigcirc p), \bigcirc\Box(\neg p + \bigcirc p), \bigcirc\Box\Diamond\neg p\}$$

and the algorithm for constructing $PDC(\alpha^*(\neg\phi))$, after stages i)-iv), gives the sets

1. $\{\Diamond p, \Diamond\neg p, (\neg p \vee \bigcirc p), \bigcirc\Box(\neg p \vee \bigcirc p), \bigcirc\Box\Diamond\neg p, p, \bigcirc\Diamond\neg p, \bigcirc p\}$
2. $\{\Diamond p, \Diamond\neg p, (\neg p \vee \bigcirc p), \bigcirc\Box(\neg p \vee \bigcirc p), \bigcirc\Box\Diamond\neg p, \bigcirc\Diamond p, \neg p\}$
3. $\{\Diamond p, \Diamond\neg p, (\neg p \vee \bigcirc p), \bigcirc\Box(\neg p \vee \bigcirc p), \bigcirc\Box\Diamond\neg p, \bigcirc\Diamond p, \neg p, \bigcirc p\}$ $\star\star\star$
4. $\{\Diamond p, \Diamond\neg p, (\neg p \vee \bigcirc p), \bigcirc\Box(\neg p \vee \bigcirc p), \bigcirc\Box\Diamond\neg p, \bigcirc\Diamond p, \bigcirc\Diamond\neg p, \neg p\}$ $\star\star\star$
5. $\{\Diamond p, \Diamond\neg p, (\neg p \vee \bigcirc p), \bigcirc\Box(\neg p \vee \bigcirc p), \bigcirc\Box\Diamond\neg p, \bigcirc\Diamond p, \bigcirc\Diamond\neg p, \bigcirc p\}$

From this list the starred sets are deleted, since they contain unnecessary formulae.

In set 3 the formulae

$$\Diamond p,\ \Diamond\neg p,\ (\neg p \vee \bigcirc p),\ \bigcirc\Box(\neg p \vee \bigcirc p),\ \text{and } \bigcirc\Box\Diamond\neg p$$

are essential, since they are components of the original formula. $\bigcirc\Diamond p$ is essential to satisfy $\Diamond p$ and $\neg p$ is essential to satisfy $\Diamond\neg p$ but $\bigcirc p$ is redundant, since $\neg p \vee \bigcirc p$ is already satisfied.

In set 4, the formula $\bigcirc\Diamond\neg p$ is redundant, since $\Diamond\neg p$ is already satisfied.

We label the remaining sets s_1, s_2, s_3 respectively.

$nexts(s_1)$ is given by

$$\{\Diamond\neg p, (\neg p \vee \bigcirc p), \bigcirc\Box(\neg p \vee \bigcirc p), \bigcirc\Box\Diamond\neg p, p\}$$

which expands to s_4, and hence s_1 has successor s_4.

$nexts(s_2)$ is given by

$$\{\Diamond p, \Diamond\neg p, (\neg p \vee \bigcirc p), \bigcirc\Box(\neg p \vee \bigcirc p), \bigcirc\Box\Diamond\neg p\}$$

which is identical to the α-expansion of $\neg\phi$ and hence s_2 has successor states s_1, s_2 and s_3

Similarly s_3 and s_4 have successors s_1 and s_4 respectively.

This gives the graph shown in figure 2.1.

All states have successors, but states s_1, s_3 and s_4 contain the formula $\Diamond\neg p$ which has the unresolvable eventuality $\neg p$.This means that these states are deleted, leaving only s_2 which now contains an unresolvable eventuality $\Diamond p$ and so is itself deleted. Since all states have been deleted the formula $\neg\phi$ is unsatisfiable and therefore ϕ is valid.

The algorithm described in [Wol83] generates a graph with 7 states for this formula.

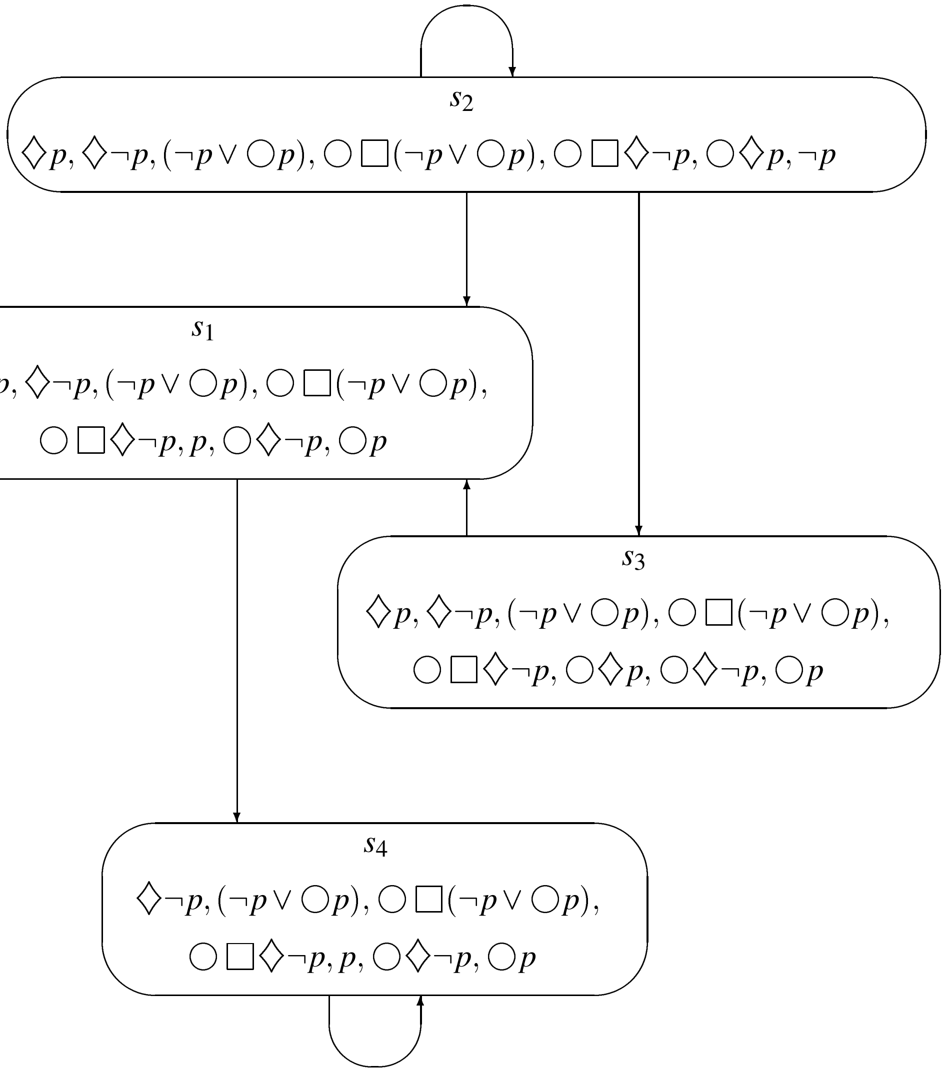

Figure 2.1: H-graph for Example 1

2.8 Example 2

Let $\psi = \Diamond p \wedge \Diamond q \wedge (p\,\mathcal{U}q)$, then we showed in section 2.4.3 that $PDC(\alpha^*(\psi))$ consists of the sets

$$\{p\,\mathcal{U}q, \Diamond q, \Diamond p, q, p\}, \{p\,\mathcal{U}q, \Diamond q, \Diamond p, q, \bigcirc\Diamond p\} \text{ and } \{p\,\mathcal{U}q, \Diamond q, \Diamond p, p, \bigcirc(p\,\mathcal{U}q), \bigcirc\Diamond q\}.$$

We create states s_1, s_2 and s_3 labelled by the above sets.

- $nexts(s_1) = \emptyset$ and so the successor state for s_1 is a state with an empty label which we denote by s_0.
- $nexts(s_2) = \{\Diamond p\}$ and so s_2 has two successor states s_4 and s_5 labelled by $\{\Diamond p, p\}$ and $\{\Diamond p, \bigcirc\Diamond p\}$ respectively.
- $nexts(s_3) = \{p\,\mathcal{U}q, \Diamond q\}$ and $PDC(nexts(s_3))$ consists of two sets $\{p\,\mathcal{U}q, \Diamond q, q\}$ and $\{p\,\mathcal{U}q, \Diamond q, p, \bigcirc(p\,\mathcal{U}q), \bigcirc\Diamond q\}$ which generate two new states s_6 and s_7.
- $nexts(s_4) = nexts(s_6) == nexts(s_1)$
- $nexts(s_5) = \{\Diamond p\} = nexts(s_2)$
- $nexts(s_7) = \{p\,\mathcal{U}q, \Diamond q\} = nexts(s_3)$.

We therefore have the graph shown in figure 2.2

All states have successors and all eventualities are resolvable, so no states are deleted from the graph. The algorithm of [Wol83] generates a graph of 26 states for this example.

Any Hintikka structure for the formula ψ has an image along a path in the graph under the map described above. For example if we take the complete Hintikka structure generated by an interpretation in which the following propositional assignments are made

$$\{p\}, \{p\}, \{p\}, \{q\}, \{p,q\}, \{p,q\}, \{p,q\}, \ldots$$

we obtain

$$\begin{aligned}
&L(m(0)) = \{p, \Diamond p, \Diamond q, p\,\mathcal{U}q, \bigcirc\Diamond p, \bigcirc\Diamond q, \bigcirc(p\,\mathcal{U}q)\}\\
&L(m(1)) = L(m(2)) = L(m(0))\\
&L(m(3)) = \{q, \Diamond p, \Diamond q, p\,\mathcal{U}q, \bigcirc\Diamond p, \bigcirc\Diamond q, \bigcirc(p\,\mathcal{U}q)\}\\
&L(m(i)) = \{p, q, \Diamond p, \Diamond q, p\,\mathcal{U}q, \bigcirc\Diamond p, \bigcirc\Diamond q, \bigcirc(p\,\mathcal{U}q)\} \text{ for } i \geq 4
\end{aligned}$$

and we have

$$\begin{aligned}
&\theta(m(0)) = s_3,\\
&\theta(m(1)) = s_7,\\
&\theta(m(2)) = s_7,\\
&\theta(m(3)) = s_6,\\
&\theta(m(i)) = s_3 \text{ for } i \geq 4
\end{aligned}$$

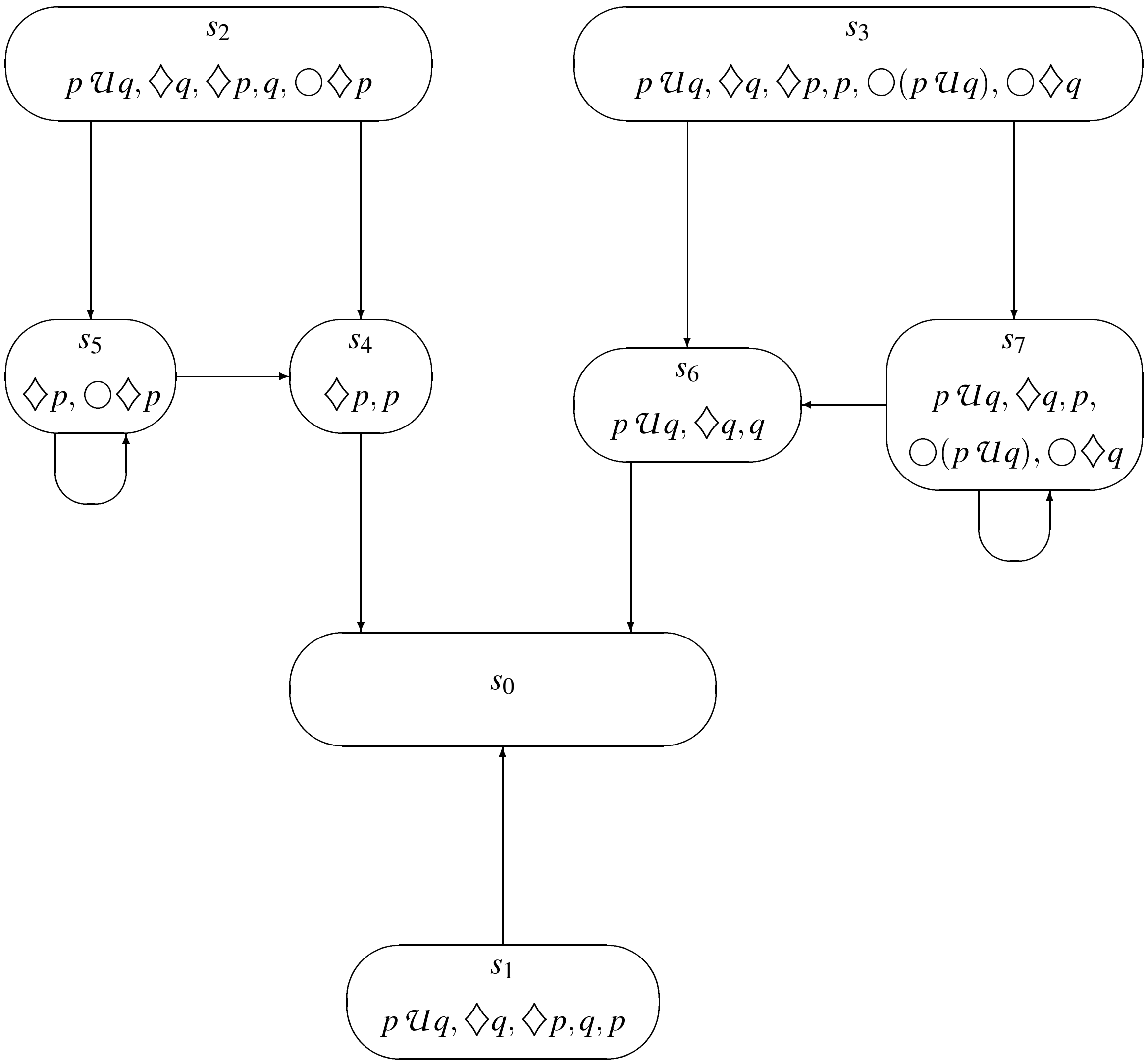

Figure 2.2: H-graph for Example 2

Chapter 3

An Axiomatisation for $\mathcal{FL}$.

In this chapter we present an axiomatisation for the language $\mathcal{FL}$ and use our decision procedure to prove that it is complete i.e. that any valid formula in $\mathcal{FL}$ is provable using our axiomatisation.

3.1 The Axiom system

The axiom system used is that given in [MP83]

Recall that $\models f$ denotes "f is valid", and define the symbol $\vdash$ by

Definition 13 $\vdash f$ *iff* f *is an axiom of* $\mathcal{FL}$ *or* f *is a theorem of* $\mathcal{FL}$.

As the basic temporal operators we use the nexttime and unless operators, $\bigcirc$ and $\mathcal{W}$ respectively. The other three operators are then defined by

3.1.1 Definitions.

D1. $\Box u$ stands for $u\,\mathcal{W}\,\mathbf{false}$.

D2. $\Diamond u$ stands for $\neg\Box\neg u$.

D3. $u\,\mathcal{U}\,v$ stands for $(u\,\mathcal{W}\,v) \wedge \Diamond v$.

3.1.2 Axioms.

There are only three axioms, namely

A1. $\vdash \neg\bigcirc w \Leftrightarrow \bigcirc\neg w$.

A2. $\vdash \bigcirc(u \Rightarrow v) \Rightarrow (\bigcirc u \Rightarrow \bigcirc v)$.

A3. $\vdash u \mathcal{W} v \Leftrightarrow (v \vee \bigcirc(u \mathcal{W} v))$.

These are supplemented by the following inference rules:-

3.1.3 Inference rules.

R1. Propositional tautology - PT

$$\vdash w \quad \text{for every } w, \text{ a substitution instance of a propositional tautology}$$

R2. Modus Ponens - MP

$$\frac{\vdash u \Rightarrow v, \ \vdash u}{\vdash v}$$

R3. $\bigcirc$ Generalisation - $\bigcirc$-G

$$\frac{\vdash u}{\vdash \bigcirc u}$$

R4. $\mathcal{W}$ Introduction - $\mathcal{W}$-I

$$\frac{\vdash w \Rightarrow (v \vee (u \wedge \bigcirc w))}{\vdash w \Rightarrow (u \mathcal{W} v)}$$

The axioms are clearly sound, so that if $\vdash f$ then $\models f$. We now show that they are complete, so that if $\models f$ then $\vdash f$.

3.2 Derived Rules and theorems

Using the above definitions, axioms and inference rules we deduce the following derived inference rules .

DR1. Propositional reasoning - PR.

$$\frac{\begin{array}{c}\vdash (u_1 \wedge u_2 \wedge u_3 \wedge \ldots \wedge un) \Rightarrow v, \\ \vdash u_1, \vdash u_2, \ldots, \vdash un\end{array}}{\vdash v}$$

Proof

This follows from the propositional tautology

$$\vdash ((u_1 \wedge u_2 \wedge \ldots \wedge un) \Rightarrow v) \Rightarrow ((u_1 \Rightarrow (u_2 \Rightarrow (\ldots \Rightarrow (un \Rightarrow v) \ldots))$$

and $n+1$ applications of modus ponens. ∎

DR2. $\bigcirc\bigcirc$ rules .

a) $$\frac{\vdash u \Rightarrow v}{\vdash \bigcirc u \Rightarrow \bigcirc v}$$

b) $$\frac{\vdash u \Leftrightarrow v}{\vdash \bigcirc u \Leftrightarrow \bigcirc v}$$

Proof

a)	1	$\vdash u \Rightarrow v$	given
	2	$\vdash \bigcirc(u \Rightarrow v)$	$\bigcirc$-G
	3	$\vdash \bigcirc u \Rightarrow \bigcirc v$	A3 and MP.

b) Follows immediately from a)

∎

DR3. $\mathcal{W}\,\mathcal{W}$ rules.

ia) $$\frac{\vdash u \Rightarrow v}{\vdash (f\,\mathcal{W} u) \Rightarrow (f\,\mathcal{W} v)}$$

ib) $$\frac{\vdash u \Leftrightarrow v}{\vdash (f\,\mathcal{W} u) \Leftrightarrow (f\,\mathcal{W} v)}$$

iia) $$\frac{\vdash u \Rightarrow v}{\vdash (u\,\mathcal{W} f) \Rightarrow (v\,\mathcal{W} f)}$$

iib) $$\frac{\vdash u \Leftrightarrow v}{\vdash (u\,\mathcal{W} f) \Leftrightarrow (v\,\mathcal{W} f)}$$

Proof

ia)	1.	$\vdash f\,\mathcal{W} u \Rightarrow u \vee (f \wedge \bigcirc(f\,\mathcal{W} u))$	A3
	2.	$\vdash u \Rightarrow v$	given
	3.	$\vdash (u \vee c) \Rightarrow (v \vee c)$	2, PT, PR
	4.	$\vdash (u \vee (f \wedge \bigcirc(f\,\mathcal{W} u))) \Rightarrow (v \vee (f \wedge \bigcirc(f\,\mathcal{W} u)))$	4, PR
	5.	$\vdash f\,\mathcal{W} u \Rightarrow (v \vee (f \wedge \bigcirc(f\,\mathcal{W} u)))$	1, 4, PR
	6.	$\vdash f\,\mathcal{W} u \Rightarrow f\,\mathcal{W} v$	$\mathcal{W}$-I

ib) follows immediately from ia).

The proof of iia) and iib) along with with proofs of the remaining derived rules and theorems of this section are of a similar nature to the above and may be found in Appendix A. ∎

As a consequence of the above we can deduce

DR4. Substitution of Equivalents.

$$\frac{\vdash a \Leftrightarrow b, \vdash f}{\vdash f[b \backslash a].}$$

where $f[b\backslash a]$ denotes the formula obtained by substituting b for every occurrence of the subformula a in f.

DR5. $\square$ Introduction - $\square$-I

$$\frac{\vdash w \Rightarrow v \wedge \bigcirc w.}{\vdash w \Rightarrow \square v.}$$

DR6. $\square$ Generalisation - $\square$-G

$$\frac{\vdash w}{\vdash \square w}$$

We now prove a series of theorems in $\mathcal{FL}$ which justify the normalisation and substitution of equivalents used in the decision procedure.

T1. $\bigcirc$ commutes with non-temporal operators.

i) $\vdash \bigcirc(u \wedge v) \Leftrightarrow (\bigcirc u \wedge \bigcirc v).$

ii) $\vdash \bigcirc(u \vee v) \Leftrightarrow (\bigcirc u \vee \bigcirc v).$

iii) $\vdash \bigcirc(u \Rightarrow v) \Leftrightarrow (\bigcirc u \Rightarrow \bigcirc v).$

T2. α, β equivalences.

i) $\vdash \square u \Leftrightarrow u \wedge \bigcirc \square u.$

ii) $\vdash \Diamond u \Leftrightarrow u \vee \bigcirc \Diamond u.$

iii) $\vdash u \,\mathcal{W}\, v \Leftrightarrow v \vee (u \wedge \bigcirc(u \,\mathcal{W}\, v)).$

iv) $\vdash u \,\mathcal{U}\, v \Leftrightarrow v \vee (u \wedge \bigcirc(u \,\mathcal{U}\, v)).$

T3. Not Equivalences.

i) $\vdash \neg(u \wedge v) \Leftrightarrow \neg u \vee \neg v.$

ii) $\vdash \neg(u \vee v) \Leftrightarrow \neg u \wedge \neg v$.

iii) $\vdash \neg\neg u \Leftrightarrow u$.

iv) $\vdash \neg \bigcirc u \Leftrightarrow \bigcirc \neg u$

v) $\vdash \neg \Box u \Leftrightarrow \Diamond \neg u$.

vi) $\vdash \neg \Diamond u \Leftrightarrow \Box \neg u$.

vii) $\vdash \neg(u\, \mathcal{U} v) \Leftrightarrow \neg v\, \mathcal{W}(\neg u \wedge \neg v)$.

viii) $\vdash \neg(u\, \mathcal{W} v) \Leftrightarrow \neg v\, \mathcal{U}(\neg u \wedge \neg v)$.

T2 and DR4 show that, given any formula f, if fn denotes the result of applying the normalisation process used in the decision procedure to f then $\vdash f$ iff $\vdash fn$ and so we can assume without loss of generality from now on that all formulae are in normalised form.

3.3 Completeness of the axiomatisation

We are now able to prove that the axiomatisation is weakly complete.

Theorem 4 *Completeness*

$$\text{If } \models u \text{ then } \vdash u.$$

We prove the result by using the decision procedure to construct an H-graph for the formula $\neg u$. Since u is valid, all states containing the formulae $\alpha^*(\neg u)$ must be eliminated by the algorithm . We show that, if a state s is eliminated, then an associated formula $af(s)$ is provable, and hence deduce that u is provable.

For any non-empty L-set T we define $af(T)$ by

$$\begin{array}{rcl} af(\{f\}) & = & \neg f \\ af(T_1 \cup T_2) & = & af(T_1) \vee af(T_2). \end{array}$$

If s is a state of an H-graph, $G = (S, L, R)$, then we abbreviate $af(L(s))$ by $af(s)$.

We now prove the theorem via a series of lemmas.

Lemma 7 *If f is an α-formula, with α-components f_1 and f_2, and if T and T_1 are given by*

$$\begin{array}{rcl} T & = & \{p_1, p_2, \ldots, p_n, f\} \\ T_1 & = & \{p_1, p_2, \ldots, p_n, f_1, f_2\} \end{array}$$

then

$$\vdash af(T) \Leftrightarrow af(T_1).$$

Proof Let $LP = \{p_1, p_2, p_3, \ldots, p_n\}$, then

$$\begin{array}{rcl} af(T) & = & af(LP) \vee \neg f \quad \text{and} \\ af(T_1) & = & af(LP) \vee \neg f_1 \vee \neg f_2. \end{array}$$

1.	$\vdash f \Leftrightarrow f_1 \wedge f_2$	T2
2.	$\vdash \neg f \Leftrightarrow \neg f_1 \vee \neg f_2$	PR
3.	$\vdash af(LP) \vee \neg f \Leftrightarrow af(LP) \vee \neg f_1 \vee \neg f_2$	PR
4.	$\vdash af(L) \Leftrightarrow af(L_1)$	

∎

This gives as an immediate corollary

Lemma 8 $\vdash af(T) \Leftrightarrow af(\alpha^*(T))$.

Lemma 9 *If f is an β-formula, with β-components f_1 and f_2, and if*

$$\begin{array}{rcl} T & = & \{p_1, p_2, \ldots, p_n, f\} \\ T_1 & = & \{p_1, p_2, \ldots, p_n, f, f_1\} \\ T_2 & = & \{p_1, p_2, \ldots, p_n, f, f_2\} \end{array}$$

then

$$\vdash af(T) \Leftrightarrow af(T_1) \wedge af(T_2).$$

Proof Similar to Lemma 7. ∎

As a consequence of Lemmas 7 and 9 we can deduce

Lemma 10 *If T is an L-set and $DC(T) = \{T_1, T_2, \ldots, T_n\}$ then*

$$\vdash (af(T_1) \wedge af(T_2) \wedge \ldots \wedge af(T_n) \Leftrightarrow af(T).$$

Lemma 11

$$\frac{T_1 \subseteq T, \vdash af(T_1)}{\vdash af(T)}$$

Proof Suppose $T = T_1 \cup T_2$

1.	$\vdash af(T) \Leftrightarrow af(T_1) \vee af(T_2)$	Defn of af
2.	$\vdash af(T_1) \Rightarrow af(T)$	PR
3.	$\vdash af(T_1)$	given
4.	$\vdash af(T)$	2, 3, MP

∎

Noting that if T is improper then $\vdash af(T)$ we obtain

Lemma 12 *If $PDC(T) = \emptyset$ then $\vdash af(T)$*

Proof If L_1 is improper then either

i) $\{p, \neg p\} \subseteq L_1$ for $p \in Prop$,

or ii) **false** $\in L_1$

In either case we have a subset L_2 of L_1 with $\vdash af(L_2)$ and therefore by lemma 11, $\vdash af(L_1)$. Lemma 3.3.4 now gives the required result. ∎

Lemma 13

$$\frac{\vdash af(nexts(s))}{\vdash af(s)}$$

Proof Let $n(s)$ denote the subset of $L(s)$ consisting of nexttime formulae. Lemma 8 and T_1 show that

$$\vdash af(nexts(s)) \Rightarrow af(n(s))$$

and Lemma 11 completes the result. ∎

Lemma 14 *If a state s in G contains a formula with unresolvable eventuality then $\vdash af(s)$.*

Proof Define, for any state t of G, the formula Nt by $Nt = \neg af(nexts(s))$. Define $[t] = \{t_1 \mid (t, t_1) \in R\}$ and $[t]^*$ by $[t]^* = \{t_1 \mid (t, t_1) \in R^*\}$. Define NNt to be the disjunction of all Nt_1's for $t_1 \in [t]^*$.

Claim that $\vdash NNs \Rightarrow \bigcirc(NNs)$. For if $u \in [s]^*$ and $PDC(nexts(u)) = \{L_1, L_2, \ldots, Ln\}$ we have

$$\vdash af(L_1) \wedge af(L_2) \wedge \ldots af(Ln) \Rightarrow af(nexts(u))$$

Now for each $Li \in PDC(nexts(u))$ either $\vdash af(Li)$ or there is a state $u_1 \in [u]$ with $L(u_1) = Li$. Therefore

$$\vdash Nu \Rightarrow \bigvee\{\neg af(v) \mid v \in [u]\}$$
$$\vdash Nu \Rightarrow \bigvee\{\neg af(n(v)) \mid v \in [u]\}$$

where $n(v) = \{\bigcirc h \in L(v)\}$, so that $\vdash af(n(v)) \Leftrightarrow \bigcirc(af(nexts(v))$, therefore

$$\vdash Nu \Rightarrow \bigvee\{\bigcirc(\neg af(nexts(v))) \mid v \in [u]\}$$
$$\vdash Nu \Rightarrow \bigcirc(\bigvee\{\neg af(nexts(v)) \mid v \in [u]\})$$
$$\vdash Nu \Rightarrow \bigcirc(\bigvee\{Nv \mid v \in [u]\}$$
$$\vdash Nu \Rightarrow \bigcirc(NNs)$$

Since this is true for all $u \in [s]^*$, we have

$$\vdash NNs \Rightarrow \bigcirc(NNs)$$

Now let s be a state in G containing a formula f with unresolvable eventuality. This means that (if f_1, f_2 denote the components of f), for all $u \in [s]^*$, $\alpha^*(f_1) \subseteq L(u)$. In fact we show that, for all $u \in [s]^*$, $\vdash Nu \Rightarrow \neg f_1$.

For $nexts(u)$ must contain f and so

$$DC(nexts(u) \cup \alpha^*(f_1)) \subseteq DC(nexts(u))$$

Therefore, since f is not satisfiable, all states labelled with elements of $DC(nexts(u) \cup \alpha^*(f_1))$ must be eliminated, and so

$$\vdash af(nexts(u) \cup \alpha^*(f_1))$$

and so

$$\vdash Nu \Rightarrow \neg f_1.$$

Therefore $\vdash NNs \Rightarrow \neg f_1$ and so by $\Box I$, $\vdash NNs \Rightarrow \Box\neg f_1$ and therefore $\vdash Ns \Rightarrow \Box\neg f_1$. But if f is a formula with eventuality f_1, $\vdash \Box\neg f_1 \Rightarrow \neg f$, and so $\vdash Ns \Rightarrow \neg f$.

Rewriting this we obtain

$$\begin{array}{l} \vdash af(nexts(s)) \vee \neg f \\ \vdash af(n(s)) \vee \neg f \\ \vdash af(n(s)\{\neg f\}) \\ \vdash af(s) \end{array}$$

∎

This completes the proof of the lemma. We now use the above results to prove the main theorem.

Proof (Theorem 4)

Since the decision procedure produces a graph in which all states containing $\alpha^*(\neg u)$ are eliminated, we have several possibilities. Either

i) Stage i) of the algorithm generates an empty graph.

ii) The first state to be eliminated from the H-graph has no successors.

or iii) The first state to be eliminated from the H-graph contains an unresolvable eventuality .

In case i) Lemma 12 ensures that $\vdash af(\alpha^*(\neg u))$ and hence, by Lemma 8 $\vdash af(\{\neg u\})$ and therefore $\vdash u$.

In cases ii) and iii) Lemmas 12 and 13 ensure that if s is the eliminated state then $\vdash af(s)$. Induction now shows that for any eliminated state s, $\vdash af(s)$, and therefore $\vdash u$. This completes the proof. ∎

Chapter 4

The Logic $\mathcal{FPL}$.

4.1 Definition of $\mathcal{FPL}$.

The language $\mathcal{FPL}$ consists of the language $\mathcal{FL}$ together with 5 new temporal operators

$\bullet$	lasttime
$\blacksquare$	heretofore (always in the past)
$\Diamond$	sometime in the past
$\mathcal{S}$	strong since
$\mathcal{Z}$	weak since

4.1.1 Syntax

The syntax of $\mathcal{FPL}$ is that of $\mathcal{FL}$ with the additional formation rule that if f, g are formulae of $\mathcal{FPL}$ then so are

$$\bullet f,\ \blacksquare f,\ \Diamond f,\ f\,\mathcal{S}\,g \text{ and } f\,\mathcal{Z}\,g$$

4.1.2 Semantics

A model M in $\mathcal{FPL}$ is a pair (m, v) where m is a denumerable set of states indexed by the integers i.e. a doubly infinite sequence

$$\ldots m_{-2}, m_{-1}, m_0, m_1, \ldots$$

and v is a map

$$v : m \to 2^{Prop}$$

as before.

An interpretation is a pair $\langle \mathcal{M}, m_i \rangle$, where m_i a state of M.

We define the semantics for the future time temporal operators as in $\mathcal{FL}$ and for the past time operators in terms of the primitive operators $\bullet$ and $\mathcal{S}$ as follows :-

$$\begin{array}{lll}\langle \mathcal{M}, m_j\rangle \models \bullet f & \text{iff} & \langle \mathcal{M}, m_{j-1}\rangle \models f \\ \langle \mathcal{M}, m_j\rangle \models f \mathcal{S} g & \text{iff} & \text{there is a } k < j \text{ with } \langle \mathcal{M}, m_k\rangle \models g \\ & & \text{and for all } i, k < i < j, \langle \mathcal{M}, m_i\rangle \models f.\end{array}$$

The semantics of the other operators are defined by the truth-functional equivalences

$$\begin{array}{rcl}\diamondsuit f & \equiv & \mathbf{true}\, \mathcal{S} f \\ \blacksquare f & \equiv & \neg \diamondsuit \neg f \\ f \mathcal{Z} g & \equiv & f \mathcal{S} g \vee \blacksquare f\end{array}$$

4.2 Semantic Properties.

As with $\mathcal{FL}$ we are able to substitute equivalents.

Lemma 15 *If f is a formula in $\mathcal{FPL}$ involving the subformula g, and h is a formula such that $g \equiv h$, then*

$$f \equiv f[h \backslash g]$$

where $f[h \backslash g]$ denotes the formula given by replacing occurences of g in f by h.

Proof

In addition to the propositional and future-time operators of $\mathcal{FL}$ we need to show that if $f, g, h \in \mathcal{FPL}$ and $f \equiv g$, then

i) $\bullet f \equiv \bullet g$

ii) $f \mathcal{S} h \equiv g \mathcal{S} h$

iii) $h \mathcal{S} f \equiv h \mathcal{S} g$

Each of the above is an immediate consequence of the definitions. ∎

We also extend the list of truth functional equivalences given in Chapter 2 to the following:-

Lemma 16 *For all $f, g, h \in \mathcal{FPL}$.*

$$
\begin{array}{rrcl}
i) & \neg\mathbf{false} & \equiv & \mathbf{true} \\
ii) & \neg\mathbf{true} & \equiv & \mathbf{false} \\
iii) & \neg\neg f & \equiv & f \\
iv) & f \wedge f & \equiv & f \\
v) & f \wedge (g \wedge h) & \equiv & (f \wedge g) \wedge h \\
vi) & f \wedge g & \equiv & g \wedge f \\
viif) & \neg \bigcirc f & \equiv & \bigcirc \neg f \\
viip) & \neg \bullet f & \equiv & \bullet \neg f \\
viiif) & \bigcirc (f \wedge g) & \equiv & \bigcirc f \wedge \bigcirc g \\
viiip) & \bullet (f \wedge g) & \equiv & \bullet f \wedge \bullet g \\
ixf) & (f \,\mathcal{U} g) & \equiv & g \vee (f \wedge \bigcirc (f \,\mathcal{U} g)) \\
ixp) & (f \,\mathcal{S} g) & \equiv & \bullet g \vee (\bullet f \wedge \bullet (f \,\mathcal{S} g)) \\
xf) & \neg (f \,\mathcal{U} g) & \equiv & \neg g \,\mathcal{W} (\neg f \wedge \neg g) \\
xp) & \neg (f \,\mathcal{S} g) & \equiv & \neg g \,\mathcal{Z} (\neg f \wedge \neg g) \\
xif) & (f \,\mathcal{W} g) & \equiv & g \vee (f \wedge \bigcirc (f \,\mathcal{W} g)) \\
xip) & (f \,\mathcal{Z} g) & \equiv & \bullet g \vee (\bullet f \wedge \bullet (f \,\mathcal{Z} g)) \\
xiif) & \neg (f \,\mathcal{W} g) & \equiv & \neg g \,\mathcal{U} (\neg f \wedge \neg g) \\
xiip) & \neg (f \,\mathcal{Z} g) & \equiv & \neg g \,\mathcal{S} (\neg f \wedge \neg g) \\
xiiif) & \Box f & \equiv & f \wedge \bigcirc \Box f \\
xiiip) & \blacksquare f & \equiv & f \wedge \bullet \blacksquare f \\
xivf) & \neg \Box f & \equiv & \Diamond \neg f \\
xivp) & \neg \blacksquare f & \equiv & \blacklozenge \neg f \\
xvf) & \Diamond f & \equiv & f \vee \bigcirc \Diamond f \\
xvp) & \blacklozenge f & \equiv & f \vee \bullet \blacklozenge f \\
xvif) & \neg \Diamond f & \equiv & \Box \neg f \\
xvip) & \neg \blacklozenge f & \equiv & \blacksquare \neg f
\end{array}
$$

The proofs of the above equivalences are almost identical to the corresponding $\mathcal{FL}$ equivalences, and we omit them.

4.2.1 Normalisation.

Again we see that any formula in $\mathcal{FPL}$ is truth-functionally equivalent to a formula in which $\neg$ is applied only to atomic propositions. We assume in the following that all formulae are in this normalised form.

4.2.2 Alpha and Beta formulae.

The classification of normalised formulae in $\mathcal{FPL}$ is into five types :-

i) Literals, i.e. propositions and their negations.

ii) Next-time formulae, i.e. formulae whose leading connective is $\bigcirc$.

iii) Last-time formulae, i.e. formulae whose leading connective is $\bullet$.

iv) α-formulae, i.e. formulae which are truth functionally equivalent to conjunctions.

v) β-formulae, i.e. formulae which are truth functionally equivalent to disjunctions.

4.2.3 α and β tables

α	α_1	α_2	β	β_1	β_2
$p \wedge q$	p	q	$p \vee q$	p	q
$\Box p$	p	$\bigcirc \Box p$	$\Diamond p$	p	$\bigcirc \Diamond p$
$\blacksquare p$	$\bullet p$	$\bullet \blacksquare p$	$\blacklozenge p$	$\bullet p$	$\bullet \blacklozenge p$
			$p\,\mathcal{U} q$	q	$p \wedge \bigcirc(p\,\mathcal{U} q)$
			$p\,\mathcal{W} q$	q	$p \wedge \bigcirc(p\,\mathcal{W} q)$
			$p\,\mathcal{S} q$	$\bullet q$	$\bullet p \wedge \bullet(p\,\mathcal{S} q)$
			$p\,\mathcal{Z} q$	$\bullet q$	$\bullet p \wedge \bullet(p\,\mathcal{Z} q)$

The definitions of the function α^* and downward closed remain as in $\mathcal{FL}$ except that they now refer to the extended α and β tables. The definition of the Fischer-Ladner closure is extended by

FLv) If $\bullet g \in FL(f)$ then $g \in FL(f)$

4.3 Eventualities.

The only new formulae with associated eventualities are $\blacklozenge h$ and $g\,\mathcal{S} h$, which both have eventuality $\bullet h$. With these extra eventualities, the definition of downward closure remains as in $\mathcal{FL}$.

4.4 Hintikka Structures.

Definition 14 *A* Hintikka structure *(over the language $\mathcal{FPL}$) is a pair H=(m, L), where m is a set of states indexed by the integers,*

$$\ldots m_{-2}, m_{-1}, m_0, m_1, \ldots$$

and

$$L\colon m \to 2^{\mathcal{FPL}}$$

is a labelling of states with formulae, which satisfies for all i

Hi) *$L(m_i)$ is proper.*

Hii) *$L(m_i)$ is downward closed.*

Hiiif) *If $\bigcirc f \in L(m_i)$ then $\alpha^*(f) \subseteq L(m_{i+1})$*

Hiiip) *If $\bullet f \in L(m_i)$ then $\alpha^*(f) \subseteq L(m_{i-1})$*

Hivf) *If $f \in L(m_i)$ with future-time eventuality e, then there is a $j \geq i$ such that $\{f\} \cup \alpha^*(e) \subseteq L(m_j)$*

Hivf) *If $f \in L(m_i)$ with past-time eventuality e, then there is a $j < i$ such that $\{f\} \cup \alpha^*(e) \subseteq L(m_j)$.*

Theorem 5 *Hintikka's Lemma for $\mathcal{FPL}$.*

A formula of $\mathcal{FL}$ is satisfiable iff there exists a (complete) Hintikka structure for f.

Once again the proof is similar to the Future time case.

4.5 H-Graphs.

We define pre-H-graphs and H-graphs as follows:-

Definition 15 *A* pre-H-graph *is a triple $G = (S, L, R)$ where*

- *S is a finite set of states,*
- *$L: S \to 2^{\mathcal{FPL}}$ is a labelling of the states,*
- *$R \subseteq S \times S$ is a relation on S*

satisfying

HGi) *For all $s \in S$, $L(s)$ is proper*

HGii) *For all $s \in S$, $L(s)$ is downward closed.*

HGiii) *For all $s \in S$, if $\bigcirc f \in L(s)$ then, for all s' with $(s, s') \in R$, $\alpha^*(f) \subseteq L(s')$*

HGiv) *For all s, if $\bullet f \in L(s)$ then, for all s' with $(s', s) \in R$, $\alpha^*(f) \subseteq L(s')$*

Definition 16 *If s is a state in the pre-H-graph $G = (S, L, R)$ and $f \in L(s)$ is a formula with future-time eventuality ev then ev is said to be* resolvable *in G if there is a state s' with $(s, s') \in R^*$ and $\{f\} \cup \alpha^*(ev) \subseteq L(s')$. (Where R^* denotes the reflexive, transitive closure of R.)*
A past-time eventuality ev of f is resolvable if there is a state s' with $(s', s) \in R^*$ and $\{f\} \cup \alpha^*(ev) \subseteq L(s')$.

Definition 17 *A pre-H-graph $G = (S, L, R)$ is an* H-graph *if it satisfies the additional properties*

HGv) *R is "both ways" total ; i.e. for all $s \in S$ there is an $s' \in S$ with $(s, s') \in R$ and an $s'' \in S$ with $(s'', s) \in R$.*

HGvi) *If $f \in L(s)$ has an eventuality ev, then ev is resolvable in G.*

As in $\mathcal{FL}$ we have the lemma

Lemma 17 *f has a Hintikka structure if and only if there exists an H-graph containing a state $s(0)$ with $\alpha^*(f) \subseteq L(s(0))$.*

Proof as in $\mathcal{FL}$.

4.6 A decision procedure for $\mathcal{FPL}$.

The decision procedure for $\mathcal{FL}$ is based on the simple idea that if a state s has a particular set of nexttime formulae, we attempt to construct new states in the future of s in which these formulae are satisfied. New states can be constructed to satisfy a given set of requirements, without having any effect on the states already constructed. We have the freedom of the prophet to predict any (reasonable) future without fear of contradiction.

The situation is markedly different when both past and future time operators are involved, because one state's future is another's past and we must maintain the equivalence $\bigcirc \bullet f \equiv f$. If we follow the straightforward constructive approach as in $\mathcal{FL}$ we may find ourselves constructing a state in the past whose future time commitments have implications for the current state which are not satisfied, since the states constructed by the downward closure method satisfy only minimal requirements. To deal with this problem we adopt a strategy of constructing minimal states as before and then constructing augmented states if and when necessary.

4.7 The algorithm.

We describe an algorithm for the construction of an H-graph, $G = (S, L, R)$ for a given formula u. In the description of the algorithm we introduce three extra relations, $R_s, R_p,$ and R_a on the set of states. Their roles are as follows:-

R_s: for any state s, $(s, s') \in R_s$ iff $L(s') \in PDC(nexts(s))$
i.e. s' is a state satisfying the minimal requirements for a successor state of s. ($nexts(s)$ is defined to be the union of the sets $\alpha^*(f)$ where $\bigcirc f \in L(s)$).

R_p: for any state s, $(s', s) \in R_s$ iff $L(s') \in PDC(lasts(s))$
i.e. s' is a state satisfying the minimal requirements for predecessor state of s. ($lasts(s)$ is defined to be the union of the sets $\alpha^*(f)$ where $\bullet f \in L(s)$).

R_a: for any state s, $(s,s') \in R_a$ iff s' has been constructed as an augmented state of s. (For all such s and s', $L(s) \subseteq L(s')$).

We also define a function

$$augs\colon S \to 2^S$$

by $augs(s) = \{s' \mid (s,s') \in R_a^*\}$, where R_a^* denotes the reflexive, transitive closure of R_a.

The algorithm is defined as follows:-

i) (Initialisation)

a) Set $S = R = R_s = R_p = R_a = \emptyset$

b) (Construct $PDC(\alpha^*(u))$). For each $c \in PDC(\alpha^*(u))$, add a new state s to S with $L(s)$ defined to be c.

ii) (Graph Construction)

a) (First approximation)

For each untreated state $s \in S$
    Construct $PDC(nexts(s))$.
    For each $c \in PDC(nexts(s))$
        Add a state s_1 with $L(s_1) = c$
        Add (s,s_1) to R_s.
        Add $\{s\} \times augs(s_1)$ to R.
    Construct $PDC(lasts(s))$.
    For each $c \in PDC(lasts(s))$
        Add a state s_1 with $L(s_1) = c$
        Add (s_1,s) to R_p.
        Add $augs(s_1) \times \{s\}$ to R.
    Mark s as treated.

b) (State augmentation)

For each unchecked pair $(s,s_1) \in R$
    if $nexts(s) \not\subseteq L(s_1)$ then
        Delete (s,s_1) from R.
        For each s_2 with $(s,s_2) \in R_s$
            if $L(s_2) \cup L(s_1)$ is proper then
                Add an s_3 with $L(s_3) = L(s_2) \cup L(s_1)$
                if $(s_1,s_3) \notin R_a$ then
                    Add (s_1,s_3) to R_a
                    For each $s4$ with $(s_1,s4) \in R$
                        Add $augs(s_3) \times \{s4\}$ to R
                    For each $s4$ with $(s4,s_1) \in R$
                        Add $\{s4\} \times augs(s_3)\}$ to R
    if $lasts(s_1) \not\subseteq L(s)$ then

Delete (s, s_1) from R.
For each s_2 with $(s_2, s_1) \in R_p$
if $L(s_2) \cup L(s)$ is proper then
Add an s_3 with $L(s_3) = L(s_2) \cup L(s)$
if $(s, s_3) \notin R_a$ then
Add (s, s_3) to R_a
For each $s4$ with $(s, s4) \in R$
Add $augs(s_3) \times \{s4\}$ to R
For each $s4$ with $(s4, s) \in R$
Add $\{s4\} \times augs(s_3)$ to R

(Note that, for a given $(s, s_1) \in R$, either
$nexts(s) \subseteq L(s_1)$, or $lasts(s_1) \subseteq L(s)$,
so that at most one of the above courses will be taken for a given (s, s_1))

if neither of the above then mark $(s, s_1) \in R$ as checked.

If at any stage of the above a state s should be found for which either $nexts(s)$ or $lasts(s)$ is improper then delete the state s from the graph.

c) Repeat a) and b) until all states in S are treated and all pairs in R are checked.

iii) (Graph contraction)
Delete all states s satisfying

a) s has no successors. i.e. there is no s_1 with $(s, s_1) \in R$.

or b) s has no predecessors. i.e. there is no s_1 with $(s_1, s) \in R$.

or c) $L(s)$ contains an eventuality ev which is not resolvable in G.

iv) Repeat iii) until no further deletions are possible.

The algorithm terminates, since stages i) and ii) construct a graph whose nodes are in one to one correspondence with a subset of $2^{FL(u)}$, and is hence finite, and stage iii) merely contracts that graph. The structure produced by the algorithm is an H-graph, since the structure generated by stages i) and ii) clearly satisfies HGi)-iv), and is therefore a pre-H-graph and stage iii) ensures that HGiv) and HGv) are satisfied. We further claim that for any satisfiable $\mathcal{FPL}$ formula u, the algorithm will construct an H-graph for u, that is that the algorithm is consistent. We justify this claim in the following section.

4.8 Consistency of the algorithm

We need to show that if u is satisfiable then the algorithm is consistent i.e. a $\mathcal{FPL}$ formula u is satisfiable iff the algorithm constructs an H-graph for u.

Proof

If u is satisfiable then we can find a complete Hintikka structure (m,V) for u. Denote by $G_0 = (S_0, L_0, R_0)$ the pre-H-graph generated by stages i) and ii) of the algorithm. As in the future time case, we construct a map

$$\theta : m \to S_0$$

such that if $S_1 = \theta(m)$ and L_1 and R_1 denote the restrictions of L_0 and R_0 to S_1, then $G_1 = (S_1, L_1, R_1)$ is an H-graph for u.

Let $F_0 \subseteq \alpha^*(u)$ be the set of all formulae in $\alpha^*(u)$ whose eventualities are satisfied at m_0, and $TE = \alpha^*(u) \cup ev(F_0)$.

Now define the set of states C_0 by

$$C_0 = \{s \mid s \in S_0 \text{ and } L_0(s) = c \in PDC(\alpha^*(u)) \text{ and } TE \subseteq c \subseteq V(m_0)\}.$$

First we observe that this set is non-empty, since $DC(TE) \subseteq DC(\alpha^*(u))$ and $V(m_0)$ is proper. We now add to C_0 all states s_1 such that $(s, s_1) \in R_a$ for some $s \in C_0$ and $TE \subseteq L_0(s_1) \subseteq V(m_0)$. Now let C_1 be the set defined by

$$C_1 = \{s \in C_0 \mid \text{there is no } s_1 \in C_0 \text{with } (s, s_1) \in R_a\}$$

Now choose $\theta(m_0)$ to be any element of C_1. (There must be at least one)

Again we now proceed inductively, but this time in both directions. Suppose we have defined $\theta(m_i)$ for $0 \leq i \leq k$. If $\theta(m_k) = s_k$ then $nexts(s_k) \subseteq nexts(m_k)$ and $nexts(m_k) \subseteq V(m_{k+1})$ therefore if F_{k+1} denotes those formulae in $nexts(s_k)$ whose eventualities are satisfied in m_{k+1} and TE denotes $nexts(s_k) \cup F_{k+1}$, we have $TE \subseteq V(m_{k+1})$.

Now define the set of states C_0 by

$$C_0 = \{s \mid s \in S_0, (s_k, s) \in R \text{ and } TE \subseteq L_0(s) \subseteq V(m_{k+1})\}.$$

We claim that this set is non-empty, a claim whose justification we postpone for a moment. Now let C_1 be defined by

$$C_1 = \{s \in C_0 \mid \text{There is no } s_1 \in C_0 \text{ with } (s, s_1) \in Ra\},$$

and choose $\theta(m_{k+1})$ to be any element of C_1.

We now prove that C_0 is non-empty. Since $DC(TE) \subseteq DC(nexts(s_k))$ and $V(m_{k+1})$ is proper, there exist states s with $L_0(s) \in PDC(TE)$ and $TE \subseteq L_0(s) \subseteq V(m_{k+1})$. In fact all such states must also satisfy $(s_k, s) \in R$, since otherwise we would have $lasts(s) \not\subseteq L_0(s_k)$, in which case the algorithm would construct a state s_2 with $L(s_k) \cup lasts(s) \subseteq L(s_2)$, and so s_2 would be an augmented state of s_k satisfying $L(s_k) \subseteq L(s_2) \subseteq V(m_k)$. But s_k was constructed so that no such state s_2 exists and therefore $(s_k, s) \in R$.

We define $\theta(m_i)$ for $i < 0$ in a similar manner replacing nexts by lasts in the above.

Now if $S_1 = \theta(m)$, and L_1, R_1 are defined as above then we claim that (S_1, L_1, R_1) is an H-graph for f and so none of the states of S_1 will be eliminated at stage iii) of the algorithm. By construction every $s \in S_1$ has a successor and predecessor, and if $f \in L_1(s)$ has future eventuality g and $s = \theta(m_i)$ then there is a $k \geq i$ with $\{f\} \cup \theta(m_k)$ then $(s, s_1) \in R^*$ and so f is resolvable. Similarly past-time eventualities are resolvable. This completes the proof. ∎

4.9 Examples

We conclude this chapter with two worked examples to illustrate the decision procedure and the construction of the map described in the previous section.

4.9.1 Example 1.

In this example we use the algorithm to build an H-graph for the formula

$$u = \Diamond\blacksquare p \wedge \Diamond\Box\neg p.$$

which is of course satisfiable. For example in the interpretation

$$\ldots p, p, p, p, p, \boxed{\neg p}, \neg p, \neg p, \neg p, \neg p, \neg p, \neg p, \ldots$$

where the boxed state is m_0.

Stages i) and iia) of the algorithm result in a graph described by the following table.

No	Label	Successors	Predecessors
1	$\Diamond\blacksquare p, \Diamond\Box\neg p, \bullet\blacksquare p, \bullet p, \neg p, \bigcirc\Box\neg p$	5	6,4
2	$\Diamond\blacksquare p, \Diamond\Box\neg p, \bullet\blacksquare p, \bullet p, \bigcirc\Diamond\Box\neg p$	7,8	6,4
3	$\Diamond\blacksquare p, \Diamond\Box\neg p, \bigcirc\Diamond\blacksquare p, \neg p, \bigcirc\Box\neg p$	9,10	4
4	$\Diamond\blacksquare p, \Diamond\Box\neg p, \bigcirc\Diamond\blacksquare p, \bigcirc\Diamond\Box\neg p$	1,2,3,4	4
5	$\neg p, \bigcirc\Box\neg p$	5	1,5,7,9
6	$p, \bullet p, \bullet\blacksquare p$	1,2,6,9	6
7	$\Diamond\Box\neg p, \neg p, \bigcirc\Box\neg p$	5	2,8
8	$\Diamond\Box\neg p, \bigcirc\Diamond\Box\neg p$	7,8	2,8
9	$\Diamond\blacksquare p, \bullet p, \bullet\blacksquare p, \neg p, \bigcirc\Box\neg p$	5	3,6,10
10	$\Diamond\blacksquare p, \bigcirc\Diamond\blacksquare p, \neg p, \bigcirc\Box\neg p$	9,10	3,10

and the relations R_s, R_p are shown in the following table

	1	2	3	4	5	6	7	8	9	10
1					s					
2							s	s		
3									s	s
4	s	s	s	s						
5					s					
6	p	p				p			p	
7					s					
8							s	s		
9					s					
10									s	s

If we complete stage ii) of the algorithm we obtain the graph described by the table

No	Label	Successors	Predecessors
1	$\Diamond\blacksquare p, \Diamond\Box\neg p, \bullet\blacksquare p, \bullet p, \neg p, \bigcirc\Box\neg p$	5	6, 11
2	$\Diamond\blacksquare p, \Diamond\Box\neg p, \bullet\blacksquare p, \bullet p, \bigcirc\Diamond\Box\neg p$	7, 8	6, 11
3	$\Diamond\blacksquare p, \Diamond\Box\neg p, \bigcirc\Diamond\blacksquare p, \neg p, \bigcirc\Box\neg p$	10	4, 11
4	$\Diamond\blacksquare p, \Diamond\Box\neg p, \bigcirc\Diamond\blacksquare p, \bigcirc\Diamond\Box\neg p$	3, 4, 11	4, 11
5	$\neg p, \bigcirc\Box\neg p$	5	1, 5, 7, 9
6	$p, \bullet p, \bullet\blacksquare p$	1, 2, 6, 9, 11	6
7	$\Diamond\Box\neg p, \neg p, \bigcirc\Box\neg p$	5	2, 8
8	$\Diamond\Box\neg p, \bigcirc\Diamond\Box\neg p$	7, 8	2, 8
9	$\Diamond\blacksquare p, \bullet p, \bullet\blacksquare p, \neg p, \bigcirc\Box\neg p$	5	6
10	$\Diamond\blacksquare p, \bigcirc\Diamond\blacksquare p, \neg p, \bigcirc\Box\neg p$	10	3, 1
11	$\Diamond\blacksquare p, \Diamond\Box\neg p, \bigcirc\Diamond\blacksquare p, \bigcirc\Diamond\Box\neg p, p, \bullet\blacklozenge, \bullet\blacksquare p$	1, 2, 3, 4, 11	4, 11, 6

The only pair in the relation R_a is $(4, 11)$.

The only formulae with eventualities are $\Diamond\blacksquare p$ and $\Diamond\Box\neg p$. The eventuality $\blacksquare p$ of $\Diamond\blacksquare p$ is resolvable in states 1, 2, 9 and 11, but not in 3, 4 and 10, so these states are deleted. The formula $\Diamond\Box\neg p$ occurs amongst the remaining states in 1, 2, 7, 8 and 11 and is resolvable in each.This leaves the final graph as shown in figure 4.1.

We now illustrate the construction of the map θ using the above example.

If we define the Hintikka structure $H = (m, V)$ by

$$\begin{array}{lcl}
V(m_i) & = & \{p, \Diamond\blacksquare p, \Diamond\Box\neg p, \bullet\blacksquare p, \bullet p, \bigcirc\Diamond\blacksquare p, \bigcirc\Diamond\Box\neg p\} \text{ for } i < 0 \\
V(m_0) & = & \{p, \Diamond\blacksquare p, \Diamond\Box\neg p, \bullet\blacksquare p, \bullet p, \bigcirc\Diamond\blacksquare p, \bigcirc\Diamond\Box\neg p\} \\
V(m_1) & = & \{p, \Diamond\blacksquare p, \Diamond\Box\neg p, \bullet\blacksquare p, \bullet p, \bigcirc\Diamond\blacksquare p, \bigcirc\Diamond\Box\neg p\} \\
V(m_2) & = & \{p, \Diamond\blacksquare p, \Diamond\Box\neg p, \bullet\blacksquare p, \bullet p, \bigcirc\Diamond\blacksquare p, \bigcirc\Box\neg p, \bigcirc\Diamond\Box\neg p\} \\
V(m_3) & = & \{\neg p, \Diamond\blacksquare p, \Diamond\Box\neg p, \bullet\blacksquare p, \bullet p, \bigcirc\Box\neg p, \bigcirc\Diamond\Box\neg p\} \\
V(m_4) & = & \{\neg p, \Diamond\Box\neg p, \bullet\blacksquare p, \bigcirc\Box\neg p, \bigcirc\Diamond\Box\neg p\} \\
V(m_5) & = & \{\neg p, \Diamond\Box\neg p, \bigcirc\Box\neg p, \bigcirc\Diamond\Box\neg p\} \\
V(m_i) & = & \{\neg p, \Diamond\Box\neg p, \bigcirc\Box\neg p, \bigcirc\Diamond\Box\neg p\} \text{ for } i > 5.
\end{array}$$

then H is a complete Hintikka structure for u.

$\alpha^*(u) = \{\Diamond\blacksquare p, \Diamond\Box\neg p\}$ and the only formula of $\alpha^*(u)$ with eventuality satisfied at m_0 is $\Diamond\blacksquare p$, and so we set $TE = \{\Diamond\blacksquare p, \Diamond\Box\neg p, \bullet p, \bullet\blacksquare p\}$ and therefore C_0 consists of the single state s_2, which has no augmented states and so $\theta(m_0) = s_2$. Similarly we see that $\theta(m_1) = s_8$, $\theta(m_2) = s_8$, $\theta(m_3) = s_7$, and $\theta(m_i) = s_5$ for $i \geq 4$.

For all $i < 0$, $\theta(m_i) = s_6$.

4.9.2 Example 2

We now use the decision procedure to prove that the formula

$$u = (\Diamond\blacksquare p \Rightarrow \blacksquare p)$$

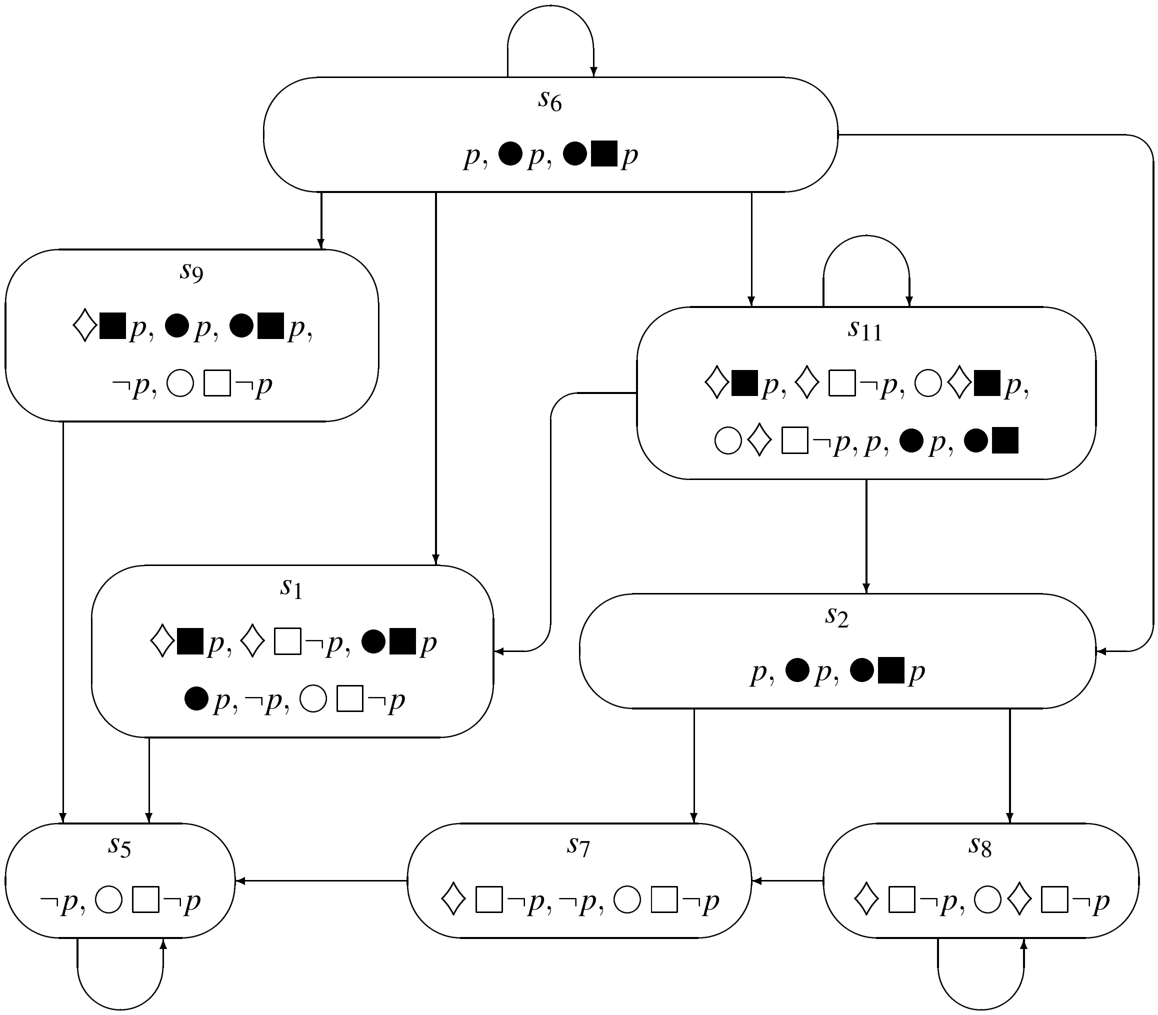

Figure 4.1: H-graph

is valid.

After normalisation $\alpha^*(\neg u) = \{\Diamond\blacksquare p, ◈\neg p\}$
Stages i) and iia) of the algorithm result in a graph described by the following table.

No	Label	Successors	Predecessors
1	$\Diamond\blacksquare p, ◈\neg p, \bullet p, \bullet\blacksquare p$		
2	$\Diamond\blacksquare p, ◈\neg p, \bullet p, \bullet\blacksquare p$		6
3	$\Diamond\blacksquare p, ◈\neg p, \bigcirc\Diamond\blacksquare p, \bullet\neg p$	7, 8	9
4	$\Diamond\blacksquare p, ◈\neg p, \bigcirc\Diamond\blacksquare p, \bullet ◈\neg p$	7, 8	10, 11
5	$p, \bullet p, \bullet\blacksquare p, ◈\neg p, \bullet\neg p$		$***$
6	$p, \bullet p, \bullet\blacksquare p, ◈\neg p, \bullet ◈\neg p$	2, 6	6
7	$\Diamond\blacksquare p, \bullet p, \bullet\blacksquare p$		12, 3, 4, 8
8	$\Diamond\blacksquare p, \bigcirc\Diamond\blacksquare p$	7, 8	3, 4, 8
9	$\neg p$	3, 10	
10	$◈\neg p, \bullet\neg p$	4, 11	9
11	$◈\neg p, \bullet ◈\neg p$	4, 11	10, 11
12	$p, \bullet p, \bullet\blacksquare p$	7, 12	12

and the relations R_s, R_p are shown in the following table

	1	2	3	4	5	6	7	8	9	10	11	12
1												
2												
3							s	s				
4							s	s				
5		p										
6		p				p						
7												
8							s	s				
9			p							p		
10				p							p	
11				p							p	
12							p					p

If we complete stage ii) of the algorithm we obtain the graph described by the table

No	Label	Successors	Predecessors
1	$\Diamond\blacksquare p, \Diamonddot\neg p, \bullet p, \bullet\blacksquare p$		$***$
2	$\Diamond\blacksquare p, \Diamonddot\neg p, \bullet p, \bullet\blacksquare p$		6
3	$\Diamond\blacksquare p, \Diamonddot\neg p, \bigcirc\Diamond\blacksquare p, \bullet\neg p$	7, 8	9
4	$\Diamond\blacksquare p, \Diamonddot\neg p, \bigcirc\Diamond\blacksquare p, \bullet\Diamonddot\neg p$	8	10, 11, 6
5	$p, \bullet p, \bullet\blacksquare p, \Diamonddot\neg p, \bullet\neg p$		$***$
6	$p, \bullet p, \bullet\blacksquare p, \Diamonddot\neg p, \bullet\Diamonddot\neg p$	2, 6, 4, 11, 14	6
7	$\Diamond\blacksquare p, \bullet p, \bullet\blacksquare p$		12, 14, 15
8	$\Diamond\blacksquare p, \bigcirc\Diamond\blacksquare p$	8	3, 4, 8, 14, 15
9	$\neg p$	3, 10	
10	$\Diamonddot\neg p, \bullet\neg p$	4, 11	9
11	$\Diamonddot\neg p, \bullet\Diamonddot\neg p$	4, 11, 14	10, 11, 6
12	$p, \bullet p, \bullet\blacksquare p$	7, 12	12
13	$\Diamond\blacksquare p, \Diamonddot\neg p, \bigcirc\Diamond\blacksquare p, \bullet\neg p, p, \bullet p, \bullet\blacksquare p$		$***$
14	$\Diamond\blacksquare p, \Diamonddot\neg p, \bigcirc\Diamond\blacksquare p, \bullet\Diamonddot\neg p, p, \bullet p, \bullet\blacksquare p$	7, 8, 15	6
15	$\Diamond\blacksquare p, \Diamonddot\neg p, \bigcirc\Diamond\blacksquare p, \bullet\neg p, p, \bullet p, \bullet\blacksquare p$	7, 8, 15	14, 15

The states 1, 5 and 13 are deleted because their lasts are improper, and states 2, 4, 8, 6 and 14 because they contain unresolvable eventualities. None of the remaining states contain $\alpha^*(\neg u)$ and so u is shown to be valid.

Chapter 5

Bounded Time Temporal Logics.

In [BKP84] compositional temporal logic proof systems for the specification and verification of concurrent programs are presented. The form of future time temporal language used in these systems has semantics defined in terms of state sequences which may be finite. In this chapter we describe a language $\mathcal{BFL}$ (Bounded $\mathcal{FL}$) which is a subset of this language and provide a decision procedure for the language by means of a translation into $\mathcal{FL}$. In [BK83] a Temporal Logic involving past time operators over a bounded past is introduced; in Appendix B a similar translation is given for this language into $\mathcal{FPL}$.

5.1 Syntax

The language $\mathcal{BFL}$ consists of

- A set *Prop* of propositions.
  (In [BKP84] these consist of two disjoint sets of state and edge propositions).
- Logical constants **true**, **false**
- Boolean connectives $\wedge, \neg, \vee, \Rightarrow, =$
- Temporal operators $\bigcirc$, $\mathcal{U}$, $\Box$, $\Diamond$

with the following formation rules for formulae in $\mathcal{BFL}$

- Any $p \in Prop$ is a formula.
- **true**, **false** are formulae.
- If f, g are formulae then so are

$$f \wedge g,\ \neg f,\ f \vee g,\ f \Rightarrow g,\ f \Leftrightarrow g,\ \bigcirc f,\ \Box f,\ \Diamond f,\ f\,\mathcal{U} g.$$

- All formulae in $\mathcal{BFL}$ are formed by the above rules.

5.2 Semantics.

A model in $\mathcal{BFL}$ is a finite or infinite state sequence

$$m = m_0, m_1, m_2, \ldots$$

together with an assignment of propositions to states

$$v : m \to 2^{Prop}$$

If $m = m_0, m_1, m_2, \ldots, m_k$ is a finite state sequence we define the length of m, $l(m)$ to be $k+1$. If m is an infinite sequence then $l(m) = \omega$, the first infinite ordinal.

The truth of a formula of $\mathcal{BFL}$ in an interpretation $\langle \mathcal{M}, m_j \rangle$ is defined as in $\mathcal{FL}$ for non-temporal operators and for temporal operators as follows

$$\begin{array}{llll}
\langle \mathcal{M}, m_j \rangle & \models & \bigcirc f & \text{iff } l(m) > j+1 \text{ and } \langle \mathcal{M}, m_{j+1} \rangle \models f \\
\langle \mathcal{M}, m_j \rangle & \models & \Box f & \text{iff } \langle \mathcal{M}, m_i \rangle \models f \text{ for all } i, j \leq i < l(m) \\
\langle \mathcal{M}, m_j \rangle & \models & \Diamond f & \text{iff } \langle \mathcal{M}, m_i \rangle \models f \text{ for some } i, j \leq i < l(m) \\
\langle \mathcal{M}, m_j \rangle & \models & f \,\mathcal{U} g & \text{iff there is a } k, j \leq k < l(m), \text{ with } \langle \mathcal{M}, m_k \rangle \models g \text{ and} \\
 & & & \text{for all } i \text{ with } ji \leq k, \langle \mathcal{M}, m_i \rangle \models f.
\end{array}$$

5.3 Decision Procedure.

To develop a decision procedure for a particular language one can either construct one specifically for that language, or provide a translation into some other language for which a decision procedure is already known. We adopt the latter approach, and give a translation into $\mathcal{FL}$.

The translation consists of a pair of maps τ_l and τ_m where

$$\tau_l : \mathcal{BFL} \to \mathcal{FL}$$
$$\tau_m : ModB \to Mod$$

and $ModB$, Mod denote the sets of models of $\mathcal{BFL}$ and $\mathcal{FL}$ respectively.

These maps have the property that

$$\begin{array}{lll}
\langle \mathcal{M}, m_0 \rangle \models f & \text{iff} & \text{there is a model} S \text{ for } \mathcal{FL} \text{ with} \\
 & & S = \tau_m(M) \text{ and } \langle \mathcal{S}, s_0 \rangle \models \tau_l(f)
\end{array}$$

This means that if $f \in \mathcal{BFL}$ then

f satisfiable iff $\tau_l(f)$ is satisfiable.

It also means that f is valid iff $\neg\tau_l(\neg f)$ is valid. Note that f valid does not mean that $\tau_l(f)$ is valid, since although $\tau_l(f)$ is satisfied in all models in the image of τ_m, it may not be satisfied in all models.

We define the translation by introducing a labelling proposition λ. We assume throughout that λ is not a proposition of $\mathcal{BFL}$ and that the propositions of $\mathcal{FL}$ consist of those of $\mathcal{BFL}$ together with λ.

We define an operator *lab* in $\mathcal{FL}$ by $lab(\lambda) = \Box(\lambda \vee \Box\neg\lambda)$. The map τ_l is then defined by

$$\tau_l(f) = lab(\lambda) \wedge \tau_{fin}(f,\lambda)$$

where τ_{fin} is defined by

$$\begin{array}{lcll}
\tau_{fin}(f,\lambda) & = & \lambda \wedge f & \text{if } f \text{ is a literal.} \\
\tau_{fin}(\Box f,\lambda) & = & \Box(\lambda \Rightarrow \tau_{fin}(f,\lambda)) & \\
\tau_{fin}(\Diamond f,\lambda) & = & \Diamond(\lambda \wedge \tau_{fin}(f,\lambda)) & \\
\tau_{fin}(\bigcirc f,\lambda) & = & \bigcirc(\lambda \wedge \tau_{fin}(f,\lambda)) & \\
\tau_{fin}(f\,\mathcal{U} g,\lambda) & = & \tau_{fin}(f,\lambda)\,\mathcal{U}\tau_{fin}(g,\lambda) & \\
\tau_{fin}(f \wedge g,\lambda) & = & \tau_{fin}(f,\lambda) \wedge \tau_{fin}(g,\lambda) & \\
\tau_{fin}(f \vee g,\lambda) & = & \tau_{fin}(f,\lambda) \vee \tau_{fin}(g,\lambda) & \\
\tau_{fin}(\neg\Box f,\lambda) & = & \neg\tau_{fin}(\Box f,\lambda) & \\
\tau_{fin}(\neg\Diamond f,\lambda) & = & \neg\tau_{fin}(\Diamond f,\lambda) & \\
\tau_{fin}(\neg\bigcirc f,\lambda) & = & \neg\tau_{fin}(\bigcirc f,\lambda) & \\
\tau_{fin}(\neg(f\,\mathcal{U} g),\lambda) & = & \neg\tau_{fin}(f\,\mathcal{U} g,\lambda) & \\
\tau_{fin}(\neg(f \wedge g),\lambda) & = & \tau_{fin}(\neg f,\lambda) \vee \tau_{fin}(\neg g,\lambda) & \\
\tau_{fin}(\neg(f \vee g),\lambda) & = & \tau_{fin}(\neg f,\lambda) \wedge \tau_{fin}(\neg g,\lambda) &
\end{array}$$

Note that $\tau_l(\neg f) \neq \neg\tau_l(f)$.

We now define the map $\tau_m\colon ModB \to Mod$. If $M = (m, vm) \in ModB$ then define $\tau_m(M) = S = (s, vs)$, where either

$$l(m) = \omega \;\text{ and } vs(s_i) = vm(m_i) \cup \{\lambda\} \text{ for all } i \geq 0 \text{ or}$$

$$l(m) = k \;\text{ and } vs(s_i) = vm(m_i) \cup \{\lambda\} \text{ for all } i, 0 \leq i < k,\ vs(s_i) = \emptyset \text{ for all } i, k \leq i.$$

We claim the following lemma

Lemma 18 *For any $f \in \mathcal{BFL}$ $\langle \mathcal{M}, m_0\rangle \models f$ iff there is a model S for $\mathcal{FL}$ with $S = \tau_m(M)$ and $\langle \mathcal{S}, s_0\rangle \models \tau_l(f)$*

Proof (Sketch)

(if) Let *Mod* be the set of models S for $\mathcal{FL}$ such that $\langle \mathcal{S}, s_0\rangle \models lab(\lambda)$. We define a map $\pi_\lambda\colon Mod \to ModB$ as follows. If $S = (s, v)$ and $\langle \mathcal{S}, s_0\rangle \models lab(\lambda)$ then $\lambda \in v(s_i)$ for all $i, 0 \leq i < k$ where $k = \omega$ or k a natural number. We construct a model $\pi_\lambda(S) = M = (m, vm)$ for $\mathcal{BFL}$ by taking a sequence m of length k with $vm(m_i) = v(s_i) - \{\lambda\}$ for all $i < k$. π_λ projects onto that part of the sequence in which λ is true.

Now if $\langle \mathcal{S}, s_0\rangle \models \tau_l(f)$, then $\langle \mathcal{S}, s_0\rangle \models lab(\lambda)$. Let $M = \pi_\lambda(S)$, then by construction $\tau_m(M) = S$, and by checking of cases it can be seen that $\langle \mathcal{M}, m_0\rangle \models f$.

(only if) If $S = \tau_m(M)$ for any $M \in ModB$, then $\langle \mathcal{S}, s_0 \rangle \models lab(\lambda)$, and if $\langle \mathcal{M}, m_0 \rangle \models f$ then we can, again by checking cases, see that $\langle \mathcal{S}, s_0 \rangle \models \tau_l(f)$.

∎

5.4 The Chop Operator.

In [BKP84] a further binary temporal operator is introduced, namely

$$\mathcal{C} \quad \text{chop or combine}$$

This operator is used to describe sequences formed by concatenation. If $M = (m, vm), N = (n, vn)$ are $\mathcal{BFL}$ models such that either

$$\begin{array}{rl} \text{i)} & l(m) = \omega \\ \text{or ii)} & l(m) = k \text{ and } vm(m_{k-1}) = vn(n_0), \end{array}$$

then we define the model $M \circ N$, called the *fusion* of M and N, by $M \circ N = R = (r, vr)$ where

- $l(r) = l(m) + l(n) - 1$
- If m is an infinite sequence then R = M,
- If m is a finite sequence with $l(m) = k$ then

$$\begin{array}{ll} vr(r_i) = vm(m_i) & \text{for } 0 \leq i < k \\ vr(r_{k+i}) = vm(r_i) & \text{for } 0 \leq i. \end{array}$$

- Otherwise $M \circ N$ is undefined.

The semantics of the chop operator is defined by

$$\langle \mathcal{M}, m_0 \rangle \models f\,\mathcal{C}\,g \text{ iff } M = N \circ R \text{ where } \langle \mathcal{N}, n_0 \rangle \models f \text{ and } \langle \mathcal{R}, r_0 \rangle \models g$$

We denote the language formed by adding the chop operator to $\mathcal{BFL}$ by $\mathcal{BFL}^+$.

Note that, with the above definition,

$$\begin{array}{rcl} \bigcirc(f\,\mathcal{C}\,g) & \equiv & (\bigcirc f)\,\mathcal{C}\,g, \\ \Diamond(f\,\mathcal{C}\,g) & \equiv & (\Diamond f)\,\mathcal{C}\,g. \end{array}$$

There appear to be no similar formulae involving $\Box$ and $\mathcal{U}$, as we show below.

An attempt to provide a translation of the chop operator into $\mathcal{FL}$ met with only limited success. We extend the translation function τ_l to include the chop operator by introducing a new labelling proposition λ_c for each occurrence of the chop operator and defining

$$\tau_{fin}(f\,\mathcal{C}\,g, \lambda) = lab(\lambda_c) \wedge \Box(\lambda_c \Rightarrow) \wedge \tau_{fin}(f, \lambda_c) \wedge \Box(\lambda_c \wedge \bigcirc\neg\lambda_c \Rightarrow \tau_{fin}(g, \lambda))$$

The proposition λ_c determines the point at which concatenation of sequences should take place. There are, however, several problems that arise from this approach; first, because the λ_c's depend on the formula f being translated rather being an inherent property of the model, as is the case for λ, it is not possible to amend the definition of τ_m to automatically generate a model in $\mathcal{FL}$ in which $\tau_l(f)$ is satisfied. This means that the proof of Lemma 18 in this extended language no longer holds. Second, because λ_c determines the point at which concatenation should take place, we need a different λ_c for each time the formula $f\,\mathcal{C}\,g$ is to be satisfied, and not for each syntactic occurrence of $f\,\mathcal{C}\,g$ in the original formula. In fact the translation of $f\,\mathcal{C}\,g$ is a Skolemised version of the quantified formula

$$\exists\lambda_c.lab(\lambda_c) \wedge \Box(\lambda_c \Rightarrow \lambda) \wedge \tau_{fin}(f,\lambda_c) \wedge \Box(\lambda_c \wedge \bigcirc\neg\lambda_c \Rightarrow \tau_{fin}(g,\lambda))$$

and the existential quantifier cannot be moved across any of the operators $\neg$, $\Box$ or $\mathcal{U}$.

The following example illustrates the problem.

Let $u = \Box((p \wedge \Diamond\neg\bigcirc\mathbf{true})\,\mathcal{C}\,q)$. This formula is satisfied in a model $M = (m, vm)$, where m is an infinite sequence with vm defined by

$$\begin{array}{rcl} vm(m_i) & = & \{p\} \text{ if } i \text{ is not a multiple of 10,} \\ vm(m_i) & = & \{p,q\} \text{ if } i \text{ is a multiple of 10,} \end{array}$$

The translation $\tau_l(u)$ is given by

$$lab(\lambda) \wedge \Box(\Rightarrow (lab(\lambda_c) \wedge \Box(\lambda_c \Rightarrow) \wedge (p \wedge \lambda_c \wedge \Diamond\neg\bigcirc\lambda_c) \wedge \Box((\lambda_c \wedge \bigcirc\neg\lambda_c) \Rightarrow q\wedge))).$$

The translated model $S = \tau_m(M) = (s, vs)$, where s is an infinite sequence, and vs is given by

$$\begin{array}{rcl} vm(m_i) & = & \{p,\lambda\} \text{ if } i \text{ is not a multiple of 10,} \\ vm(m_i) & = & \{p,q,\lambda\} \text{ if } i \text{ is a multiple of 10,} \end{array}$$

There is no way in which we could add the single proposition λ_c to this model to give a model in which $\tau_l(u)$ is satisfied. A similar problem arises with attempts to translate the formula $\neg(f\,\mathcal{C}\,g)$. However, we do have the following limited result.

Lemma 19 *If u is a $\mathcal{BFL}^+$ formula in which $\mathcal{C}$ occurs outside the scope of temporal operators and $\neg$, then u is satisfiable iff $\tau_l(u)$ is satisfiable.*

Proof It is sufficient to prove the result for the formula u = f $\mathcal{C}$ g, since a simple induction on the structure of u then gives the required result.

(if) Suppose there exists an $\mathcal{FL}$ model $S = (s, vs)$, with $\langle\mathcal{S}, s_0\rangle \models \tau_l(f\,\mathcal{C}\,g)$, then let $M = \pi_\lambda(S)$, where π_λ is the map defined in the proof of Lemma 18, except that any λ_c's appearing in S are forgotten in M. We need to show that $\langle\mathcal{M}, m_0\rangle \models f\,\mathcal{C}\,g$, i.e. that there exist P, R with $M = P \circ R$ and

$$\langle\mathcal{P}, p_0\rangle \models f\,\langle\mathcal{R}, r_0\rangle \models g.$$

Let the map $\pi_{\lambda c}$ be defined similarly to π_λ and let $P = \pi_{\lambda c}(S)$, then, by lemma 18, since

$$\langle\mathcal{S}, s_0\rangle \models lab(\lambda_c) \wedge fin(f,\lambda),$$

we have $\langle \mathcal{P}, p_0 \rangle \models f$. If p is an infinite sequence the result is proved, otherwise the sequence p has length $k+1$ and for all $i \leq k+1$, $\neg\lambda_c \in vs(s_i)$. Now define the model $R_1 = (r1, vr1)$ to be the sequence $r1_0, r1_1, \ldots$ with $vr1$ defined by $vr1(r1_i) = vs(m_{i+k})$ for $i \geq 0$. Then $\langle \mathcal{R}_\infty, r1_0 \rangle \models lab(\lambda) \wedge \lambda \wedge \bigcirc\neg\lambda$ (by construction) and hence $\langle \mathcal{R}_\infty, r1_0 \rangle \models lab(\lambda) \wedge \tau_{fin}(g, \lambda)$, and therefore if $R = \pi_\lambda(R1)$, we have $\langle \mathcal{R}, r_0 \rangle \models g$. Now $vp(p_k) = vr(r_k)$ so if $M = P \circ R$, $\langle \mathcal{M}, m_0 \rangle \models f \, \mathcal{C} \, g$.

(only if) Suppose $\langle \mathcal{M}, m_0 \rangle \models f \, \mathcal{C} \, g$, then let $S = \tau_m(M)$. We must have $M = P \circ R$ with $\langle \mathcal{P}, p_0 \rangle \models f$ and $\langle \mathcal{R}, r_0 \rangle \models g$. Either i) p is infinite and $P = M$, or ii) p is finite.

In case i) we have $\langle \mathcal{M}, m_0 \rangle \models f$ and therefore $\langle \mathcal{S}, s_0 \rangle \models \tau_l(f)$ by lemma 18, so $\langle \mathcal{S}, s_0 \rangle \models lab(\lambda) \wedge \tau_{fin}(f, \lambda)$. We need to show that that we can add λ_c's to S so that

$$\langle \mathcal{S}, s_0 \rangle \models \Box(\lambda_c \Rightarrow) \wedge lab(\lambda_c) \wedge \Box((\lambda_c \wedge \bigcirc\neg\lambda_c) \Rightarrow \tau_{fin}(g, \lambda).$$

If we add λ_c to every $vs(s_i)$, this is clearly the case.

In case ii) we have $M = P \circ R$ where p is finite. Let $k+1 = l(p)$, then for all $i, 0 \leq i \leq k$ add λ_c to $vs(s_i)$, and for all $i, i > k$, add $\neg\lambda_c$ to $vs(s_i)$. Then, with these additional propositions,

$$\begin{array}{lcl}
\langle \mathcal{S}, s_0 \rangle & \models & lab(\lambda) \\
\langle \mathcal{S}, s_0 \rangle & \models & lab(\lambda_c) \wedge \tau_{fin}(f, \lambda_c), \text{ by Lemma 18} \\
\langle \mathcal{S}, s_0 \rangle & \models & \Box(\lambda_c \Rightarrow), \text{ by construction} \\
\langle \mathcal{S}, s_0 \rangle & \models & \Box((\lambda_c \wedge \bigcirc\neg\lambda_c) \Rightarrow \tau_{fin}(g, \lambda), \text{ since } \langle \mathcal{R}, r_0 \rangle \models g.
\end{array}$$

This proves the result. ∎

This result is somewhat limited, but is about as good as as one could expect since all decision procedures known to the author for logics containing chop-like operators are non-elementary. The result is not without value however, since the main role played by the chop operator used in [BKP84] is in the sequential composition rule

$$\frac{\vdash \{S_1\}\phi_1 \qquad \vdash \{S_2\}\phi_2}{\vdash \{S_1; S_2\}\phi_1 \, \mathcal{C} \, \phi_2}$$

To show that $S_1; S_2$ satisfies a required specification ϕ_3, we need to show that

$$\models \phi_1 \, \mathcal{C} \, \phi_2 \Rightarrow \phi_3$$

which is a formula satisfying the necessary restrictions.

Chapter 6

Implementation.

Both the future-time and past-time decision procedures described in chapters 2 and 4 have been implemented on a VAX 11/750; in this chapter we describe the basic design and data structures used in the implementation. Unless indication is made to the contrary, all remarks in this chapter apply to the implementation of the past time algorithm. The program consists of 2700 lines of Standard Pascal.

The program accepts as input a formula in $\mathcal{FL}$ or $\mathcal{FPL}$ which is to be checked for satisfiability or validity.

The three main tasks are

i) Parsing the input formula,

ii) Generation of a formula key table, and

iii) Generation of the H-graph.

6.1 Parsing.

A double priority operator precedence technique [Gri71] is used to produce the parsed formula tree.

6.2 Formula Key Generation

This stage of the program generates a table containing entries of the form shown in figure 6.1

There is one entry for each formula in FL(u) and the table is indexed by Formula Id. Formula Type distinguishes between α, β, nexttime, lasttime and literal formulae, and the SubIds are pointers to the α orβ components of the formula or, in the case of next and lasttime formulae, to the subformulae given by stripping off the leading connective.

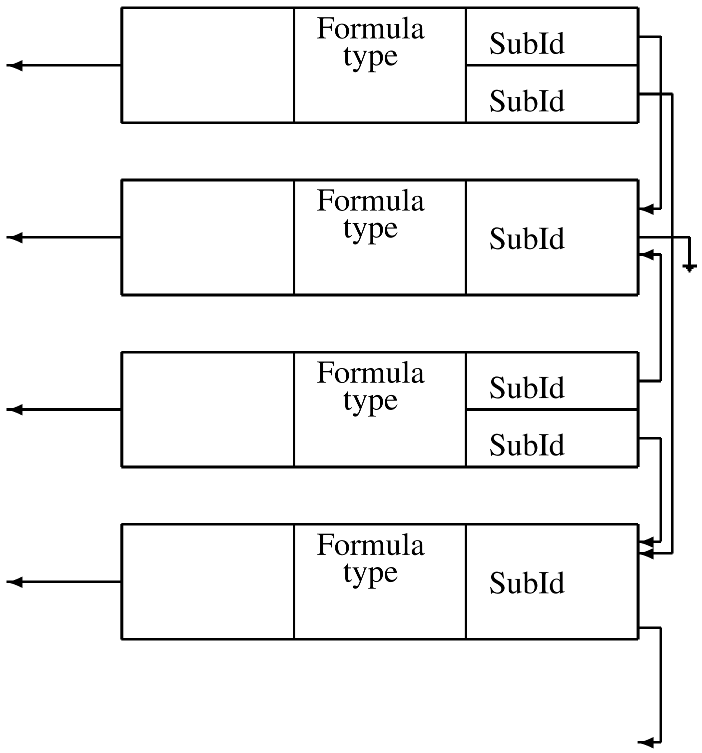

Figure 6.1: Formula table

6.3 H-graph construction.

The two main data structures used in the generation of the final H-graph are

i) The H-graph itself.

ii) A list of Pre-States.

6.3.1 The H-graph.

The H-graph consists of a set of States, which are objects of the form

Label Ptr
Succs Ptr
Preds Ptr
Augs Ptr

Label is set of Formula Ids, and Succs, Preds and Augs are sets of pointers to States.

6.3.2 Pre-States.

All States of the H-graph are labelled with L-sets which are generated by the downward closure construction from a non-closed L-set such as nexts(s), lasts(s) or $\alpha^*(u)$. We refer to such non-closed L-sets as pre-state labels.

A Pre-State is an object of the form

Label Ptr
States Ptr

A list of processed pre-states is maintained with, for each pre-state ps, pointers into the main H-graph to the states labelled by members of PDC(ps). This list is searched before any pre-state is processed.

6.4 Performance

In [Pla83a] three formulae are given as examples of formulae processed by a tableau based decision procedure implemented in InterLisp on an F2 computer. The formulae, in Plaisted's notation, are

$$\begin{array}{ll} R3: & LUA(a,x) \wedge LUA(a,y) \Rightarrow LUA(a,x \wedge y) \\ R4: & LUA(a,b \wedge c) \wedge LUA(b,a \wedge \neg c) \Rightarrow LUA(a \wedge b, false) \\ R5: & LUA(a,b) \wedge LUA(b,c) \Rightarrow LUA(a \vee b, c) \end{array}$$

where LUA is the "latches until after" operator. The formula $LUA(a,b)$ is equivalent to the $\mathcal{FL}$ formula

$$(\neg a\, \mathcal{W}(a\, \mathcal{W}(a \wedge b))$$

The timings (in seconds) for proving the validity of these formulae by the two decision procedures are, (Plaisted's timings first)

R3	81	76
R4	127	83
R5	18.8	40

It has not been possible to account for the differences in these timings, since insufficient details are given in [Pla83a] of the algorithm used.

In Appendix C we give sample output and timings for a range of formulae involving both past and future time operators.

Chapter 7

Other decision procedures.

The decision procedures for various types of linear and branching time Temporal Logics described in the literature [Wol83, EH82, MW84, Pla83b], fall into two main categories

i) Tableau based techniques.

ii) Maximal model techniques.

7.1 Tableau based techniques.

The tableau method is a bottom-up method, which for a given formula u, tries to construct, state by state, a "model" for u. The states are constructed to have the minimum properties necessary for the model. Tableau based decision procedures for $\mathcal{FL}$ and $\mathcal{FPL}$ have been discussed in some detail in previous chapters; similar procedures have been developed for branching time logics in [BAMP81, CES86], although the algorithm described in [BAMP81] has since been found to pronounce some satisfiable formulae unsatisfiable.

7.2 Maximal model techniques.

Maximal model techniques, which can be characterised as top-down, construct a model for a given formula by constructing a set of states which are maximal, in a sense to be described below, defining a relation on that set, and finally eliminating states to produce an H-graph for the formula. The algorithm described below is essentially that given in [EH82] for the branching time logic CTL, amended for the linear time logic $\mathcal{FL}$.

Since this method does not use normalised formulae, we need to extend the α, βtables to include negated formulae. The additional entries are

α	α_1	α_2	β	β_1	β_2
$\neg\neg p$	p	p	$\neg(p \wedge q)$	$\neg p$	$\neg q$
$\neg(p \vee q)$	$\neg p$	$\neg q$	$\neg\Box p$	$\neg p$	$\bigcirc\neg\Box p$
$\neg\bigcirc p$	$\bigcirc\neg p$	$\bigcirc\neg p$			
$\neg\Diamond p$	$\neg p$	$\bigcirc\neg\Diamond p$			
$\neg(p\,\mathcal{U}q)$	$\neg q$	$\neg p \vee \bigcirc\neg(p\,\mathcal{U}q)$			
$\neg(p\,\mathcal{W}q)$	$\neg q$	$\neg p \vee \bigcirc\neg(p\,\mathcal{W}q)$			

We define the extended Fischer-Ladner closure $FL^+(u)$ of a formula u by adding the additional closure property

FL^+) If $f \in FL^+(u)$, where f is not of the form $\neg g$ then $\neg f \in FL^+(u)$

The definition of downward closure is amended to refer to these new tables, and the definition of propriety to exclude a set s containing both the formulae f and $\neg f$ for any $f \in \mathcal{FL}$.

A set $s \subseteq FL^+(u)$ is said to be *maximal* if, for all formulae of the form $\neg q \in FL^+(u)$, either $\neg q \in s$ or $q \in s$ (but not both). The algorithm is then as follows

i) (State construction) Construct the set $S_0 = \{s \mid s \subseteq FL^+(u), s \text{ maximal}\}$, and the labelling map L defined by $L(s) = s$.
Define the relation R on S_0 by $(s,t) \in R$ iff

a) if $\bigcirc p \in s$ then $p \in t$ and

b) if $\neg\bigcirc p \in s$ then $\neg p \in t$

This procedure will generate many "impossible" states; for example, if $u = p\,\mathcal{U}q$, it will generate states such as $\{p\,\mathcal{U}q, \neg p, \neg q, \neg\bigcirc(p\,\mathcal{U}q)\}$. The next stage, therefore is to eliminate some of these states.

ii) (First Graph reduction) Construct the set $S_1 = \{s \in S_0 \mid s \text{ downward closed}\}$

The remainder of the procedure follows as in the tableau method

iii) (Ensure that R is total) Delete all $s \in S_1$ with no R-successors to give S_2

iv) (Check eventualities) Delete all $s \in S_2$ which contain unresolvable eventualities.

The resulting structure is an H-graph for u.

This algorithm has a worst-case running time of $2^{c*deg(u)}$, for some constant $c > 0$, but this is also its best-case running time since step i) involves constructing $2^{fl/2}$ of the subsets of $FL^+(u)$ (where $fl = |FL^+(u)|$), whatever the formula u. Tableau based methods do not suffer from this disadvantage and generally perform better in practice than maximal model methods.

7.3 Other methods.

All of the decision procedures described above involve the construction and proof of a different decision procedure for each variant of the temporal language used. A third approach, introduced by Plaisted [Pla83b], which apparently circumvents this problem, is to design a low-level language which is expressive enough to allow easy translations into the language of each temporal language of interest. The decision procedure of the low level language can then be used to in each application. In [Pla83b] it is used to obtain a decision procedure for the interval logic of Schwartz, Melliar-Smith and Vogt [SMSVP83]. The low level language itself is a generalisation of regular expressions, and its decision procedure is non-elementary.

This approach suffers from the disadvantage that, by translating into a language that is, of necessity, at least as expressive as the the original language, one introduces a level of complexity into the decision procedure which may be unnecessary.Even to show that an $\mathcal{FL}$ formula as simple as $\Box p \wedge \Diamond \neg p$ is not satisfiable using the translation involves the construction of two graphs, one with 7 nodes and 9 edges, the other with 2 nodes and 2 edges. These graphs are then merged to give the final graph of 9 nodes and 14 edges. For the same example the decision procedure described in chapter 2 creates a graph with a single state. Other, still very simple, formulae rapidly go beyond the reach of hand computation.

The dangers of combinatorial explosion are so endemic in the design of decision procedures that it seems essential to use as much semantic information peculiar to the individual language as possible in order to produce a decision procedure that can deal with anything other than trivial formulae.

Appendix A

Proofs from An Axiomatisation for $\mathcal{FL}$.

A.1 The Axiomatisation

Recall the given axiomatisation.

A.1.1 Definitions.

D1.	$\Box u$	stands for	$u\,\mathcal{W}\,\mathbf{false}$.
D2.	$\Diamond u$	stands for	$\neg\Box\neg u$.
D3.	$u\,\mathcal{U}\,q$	stands for	$(u\,\mathcal{W}\,q) \wedge \Diamond q$.

A.1.2 Axioms.

A1. $\vdash \neg\bigcirc w \Leftrightarrow \bigcirc\neg w$

A2. $\vdash \bigcirc(u \Rightarrow v) \Rightarrow (\bigcirc u \Rightarrow \bigcirc v)$

A3. $\vdash u\,\mathcal{W}\,v \Leftrightarrow (v \vee \bigcirc(u\,\mathcal{W}\,v))$

A.1.3 Inference rules.

R1. Propositional tautology - PT

$$\vdash w \quad \text{for every } w, \text{ a substitution instance of a propositional tautology}$$

R2. Modus Ponens - MP

$$\frac{\vdash u \Rightarrow v, \vdash u}{\vdash v}$$

R3. $\bigcirc$ Generalisation - $\bigcirc$-G

$$\frac{\vdash u}{\vdash \bigcirc u}$$

R4. $\mathcal{W}$ Introduction - $\mathcal{W}$-I

$$\frac{\vdash w \Rightarrow (v \vee (u \wedge \bigcirc w))}{\vdash w \Rightarrow (u \,\mathcal{W}\, v)}$$

A.2 Proofs

We now provide the a complete set of proofs for the results of Chapter 3.

DR1. Propositional reasoning - PR.

$$\frac{\vdash (u_1 \wedge u_2 \wedge u_3 \wedge \ldots \wedge un) \Rightarrow v, \quad \vdash u_1, \vdash u_2, \ldots, \vdash un}{\vdash v}$$

Proof

This follows from the propositional tautology

$$\vdash ((u1 \wedge u2 \wedge \ldots \wedge un) \Rightarrow v) \Rightarrow ((u1 \Rightarrow (u2 \Rightarrow (\ldots \Rightarrow (un \Rightarrow v) \ldots))$$

and $n+1$ applications of modus ponens. ∎

DR2. $\bigcirc\bigcirc$ rules

a) $\dfrac{\vdash u \Rightarrow v}{\vdash \bigcirc u \Rightarrow \bigcirc v}$

b) $\dfrac{\vdash u \Leftrightarrow v}{\vdash \bigcirc u \Leftrightarrow \bigcirc v}$

Proof

a)	1	$\vdash u \Rightarrow v$	given
	2	$\vdash \bigcirc(u \Rightarrow v)$	$\bigcirc$-G
	3	$\vdash \bigcirc u \Rightarrow \bigcirc v$	A3 and MP.

b) Follows immediately from a)

∎

DR3. $\mathcal{W}\ \mathcal{W}$ rules.

ia) $$\frac{\vdash u \Rightarrow v}{\vdash (f\,\mathcal{W}u) \Rightarrow (f\,\mathcal{W}v)}$$

ib) $$\frac{\vdash u \Leftrightarrow v}{\vdash (f\,\mathcal{W}u) \Leftrightarrow (f\,\mathcal{W}v)}$$

iia) $$\frac{\vdash u \Rightarrow v}{\vdash (u\,\mathcal{W}f) \Rightarrow (v\,\mathcal{W}f)}$$

iib) $$\frac{\vdash u \Leftrightarrow v}{\vdash (u\,\mathcal{W}f) \Leftrightarrow (v\,\mathcal{W}f)}$$

Proof

ia)	1.	$\vdash f\,\mathcal{W}u \Rightarrow u \vee (f \wedge \bigcirc(f\,\mathcal{W}u))$	A3
	2.	$\vdash u \Rightarrow v$	given
	3.	$\vdash (u \vee c) \Rightarrow (v \vee c)$	2, PT, PR
	4.	$\vdash (u \vee (f \wedge \bigcirc(f\,\mathcal{W}u))) \Rightarrow (v \vee (f \wedge \bigcirc(f\,\mathcal{W}u)))$	3, PR
	5.	$\vdash f\,\mathcal{W}u \Rightarrow (v \vee (f \wedge \bigcirc(f\,\mathcal{W}u)))$	1, 4, PR
	6.	$\vdash f\,\mathcal{W}u \Rightarrow f\,\mathcal{W}v$	$\mathcal{W}$-I
iia)	1.	$\vdash u\,\mathcal{W}f \Rightarrow f \vee (u \wedge \bigcirc(u\,\mathcal{W}f))$	A3.
	2.	$\vdash u \Rightarrow v$	given
	3.	$\vdash (u \wedge c) \Rightarrow (v \wedge c)$	2, PT, PR
	4.	$\vdash (f \vee (u \wedge c)) \Rightarrow (f \vee (v \wedge c))$	3, PT, PR
	5.	$\vdash (f \vee (u \wedge \bigcirc(u\,\mathcal{W}f))) \Rightarrow (f \vee (v \wedge \bigcirc(u\,\mathcal{W}f)))$	4, PR
	6.	$\vdash u\,\mathcal{W}f \Rightarrow (v \vee (f \wedge \bigcirc(u\,\mathcal{W}f)))$	1, 5, PR
	7.	$\vdash u\,\mathcal{W}f \Rightarrow v\,\mathcal{W}f$	$\mathcal{W}$-I

ib) and iib) are immediate consequences of ia) and iia). ∎

DR4. Substitution of Equivalents.

$$\frac{\vdash a \Leftrightarrow b, \vdash f}{\vdash f[b \backslash a].}$$

Proof

Immediate from DR2, DR3 and PT. ∎

DR5. $\square$ Introduction - $\square$-I

$$\frac{\vdash w \Rightarrow v \wedge \bigcirc w.}{\vdash w \Rightarrow \square v.}$$

Proof

1.	$\vdash w \Rightarrow v \wedge \bigcirc w.$	given
2.	$\vdash w \Rightarrow false \vee (v \wedge \bigcirc w)$	1, PR.
3.	$\vdash w \Rightarrow v \mathcal{W} false.$	$\mathcal{W}$-I.
4.	$\vdash w \Rightarrow \square v$	D1.

∎

DR6. $\square$ Generalisation - $\square$-G

$$\frac{\vdash w}{\vdash \square w}$$

Proof

1.	$\vdash w$	given
2.	$\vdash \bigcirc w$	$\bigcirc$-G
3.	$\vdash w \wedge \bigcirc w$	PR
4.	$\vdash w \Rightarrow w \wedge \bigcirc w$	PR
5.	$\vdash w \Rightarrow \square w$	$\square$-I
6.	$\vdash \square w$	1, 5, MP

∎

T1. $\bigcirc$ commutes with non-temporal operators.

i) $\vdash \bigcirc(u \wedge v) \Leftrightarrow (\bigcirc u \wedge \bigcirc v).$

ii) $\vdash \bigcirc(u \vee v) \Leftrightarrow (\bigcirc u \vee \bigcirc v).$

iii) $\vdash \bigcirc(u \Rightarrow v) \Leftrightarrow (\bigcirc u \Rightarrow \bigcirc v).$

Proof

i)	1.	$u \Rightarrow (v \Rightarrow (u \wedge v))$	PT
	2.	$\bigcirc u \Rightarrow \bigcirc(v \Rightarrow (u \wedge v))$	$\bigcirc\bigcirc$
	3.	$\bigcirc(v \Rightarrow (u \wedge v)) \Rightarrow (\bigcirc v \Rightarrow \bigcirc(u \wedge v))$	A2
	4.	$\bigcirc u \Rightarrow (\bigcirc v \Rightarrow \bigcirc(u \wedge v))$	2, 3, PR
	5.	$(\bigcirc u \wedge \bigcirc v) \Rightarrow \bigcirc(u \wedge v)$	4, PR
	6.	$u \wedge v \Rightarrow u$	PT
	7.	$\bigcirc(u \wedge v) \Rightarrow \bigcirc u$	$\bigcirc$-G
	8.	$u \wedge v \Rightarrow v$	PT
	9.	$\bigcirc(u \wedge v) \Rightarrow \bigcirc v$	$\bigcirc$-G
	10.	$\bigcirc(u \wedge v) \Rightarrow \bigcirc u \wedge \bigcirc v$	7, 9, PR
	11.	$\bigcirc(u \wedge v) \Leftrightarrow \bigcirc u \wedge \bigcirc v$	5, 10

ii) and iii) follow from i) and A2.

∎

T2. α, β equivalences.

i) $\vdash \Box u \Leftrightarrow u \wedge \bigcirc \Box u$.

ii) $\vdash \Diamond u \Leftrightarrow u \vee \bigcirc \Diamond u$.

iii) $\vdash u \mathcal{W} v \Leftrightarrow v \vee (u \wedge \bigcirc(u \mathcal{W} v))$.

iv) $\vdash u \mathcal{U} v \Leftrightarrow v \vee (u \wedge \bigcirc(u \mathcal{U} v))$.

Proof

i)	1.	$\vdash \Box u \Leftrightarrow u \mathcal{W} false$	D1
	2.	$\vdash \Box u \Leftrightarrow false \vee (u \wedge \bigcirc(u \mathcal{W} false))$	A3
	3.	$\vdash \Box u \Leftrightarrow u \wedge \bigcirc(u \mathcal{W} false)$	PR
	4.	$\vdash \Box u \Leftrightarrow u \wedge \bigcirc(\Box u)$	D1.
ii)	1.	$\vdash \Diamond u \Leftrightarrow \neg \Box \neg u$	D2
	2.	$\vdash \Diamond u \Leftrightarrow \neg(\neg u \wedge \bigcirc \Box \neg u)$	i), PR
	3.	$\vdash \Diamond u \Leftrightarrow u \vee \neg \bigcirc \Box \neg u$	PT, PR
	4.	$\vdash \Diamond u \Leftrightarrow u \vee \bigcirc \neg \Box \neg u$	A1, DR4
	5.	$\vdash \Diamond u \Leftrightarrow u \vee \bigcirc \Diamond u$.	D2.

iii) is A3 and iv) follows from D3, ii) and iii).

∎

T3. Not Equivalences.

i) $\vdash \neg(u \wedge v) \Leftrightarrow \neg u \vee \neg v.$

ii) $\vdash \neg(u \vee v) \Leftrightarrow \neg u \wedge \neg v.$

iii) $\vdash \neg\neg u \Leftrightarrow u.$

iv) $\vdash \neg \bigcirc u \Leftrightarrow \bigcirc \neg u$

v) $\vdash \neg \Box u \Leftrightarrow \Diamond \neg u.$

vi) $\vdash \neg \Diamond u \Leftrightarrow \Box \neg u.$

vii) $\vdash \neg(u\, \mathcal{U}\, v) \Leftrightarrow \neg v\, \mathcal{W}\, (\neg u \wedge \neg v).$

viii) $\vdash \neg(u\, \mathcal{W}\, v) \Leftrightarrow \neg v\, \mathcal{U}\, (\neg u \wedge \neg v).$

Proof

i), ii) and iii) are all PT's, iv) is an axiom and v), vi) follow immediately from iii) and D2.

vii)	1.	$\vdash \neg(u\, \mathcal{U}\, v) \Rightarrow \neg(v \vee (u \wedge \bigcirc(u\, \mathcal{U}\, v)))$	A3, PR
	2.	$\vdash \neg(u\, \mathcal{U}\, v) \Rightarrow \neg v \wedge (\neg u \vee \neg \bigcirc(u\, \mathcal{U}\, v)))$	PR
	3.	$\vdash \neg(u\, \mathcal{U}\, v) \Rightarrow (\neg v \wedge \neg u) \vee (\neg v \wedge \bigcirc \neg(u\, \mathcal{U}\, v)))$	PR, A1
	4.	$\vdash \neg(u\, \mathcal{U}\, v) \Rightarrow \neg v\, \mathcal{W}\, (\neg u \wedge \neg v)$	$\mathcal{W}$-I
	5.	$\vdash (\neg v\, \mathcal{W}\, (\neg u \wedge \neg v)) \wedge (u\, \mathcal{W}\, v) \Leftrightarrow$ $(((\neg u \wedge \neg v) \vee (\neg v \wedge \bigcirc(\neg v\, \mathcal{W}\, (\neg u \wedge \neg v))) \wedge (v \vee (u \wedge \bigcirc(u\, \mathcal{W}\, v)))$	A3, PR
	6.	$\vdash ((\neg v\, \mathcal{W}\, (\neg u \wedge \neg v)) \wedge (u\, \mathcal{W}\, v) \Leftrightarrow$ $((\neg v \wedge \bigcirc(\neg v\, \mathcal{W}\, (\neg u \wedge \neg v)) \wedge (u \wedge \bigcirc(u\, \mathcal{W}\, v)))$	PR
	7.	$\vdash ((\neg v\, \mathcal{W}\, (\neg u \wedge \neg v)) \wedge (u\, \mathcal{W}\, v) \Leftrightarrow$ $((u \wedge \neg v \wedge \bigcirc(\neg v\, \mathcal{W}\, (\neg u \wedge \neg v) \wedge (u \wedge \bigcirc(u\, \mathcal{W}\, v)))$	PR
	8.	$\vdash ((\neg v\, \mathcal{W}\, (\neg u \wedge \neg v)) \wedge (u\, \mathcal{W}\, v) \Rightarrow \Box(u \wedge \neg v)$	$\Box$-I
	9.	$\vdash u \wedge \neg v \Rightarrow \neg v$	PT
	10.	$\vdash \Box(u \wedge \neg v) \Rightarrow \Box(\neg v)$	$\Box$-G
	11.	$\vdash ((\neg v\, \mathcal{W}\, (\neg u \wedge \neg v)) \wedge (u\, \mathcal{W}\, v) \Rightarrow \Box(\neg v)$	PR
	12.	$\vdash ((\neg v\, \mathcal{W}\, (\neg u \wedge \neg v)) \wedge (u\, \mathcal{W}\, v) \Rightarrow \neg \Diamond(v)$	D1, PR
	13.	$\vdash (a \wedge b \Rightarrow \neg c) \Rightarrow (a \Rightarrow \neg(b \wedge c))$	PT
	14.	$\vdash (\neg v\, \mathcal{W}\, (\neg u \wedge \neg v)) \Rightarrow \neg((u\, \mathcal{W}\, v) \wedge \Diamond v)$	PR
	15.	$\vdash (\neg v\, \mathcal{W}\, (\neg u \wedge \neg v)) \Rightarrow \neg(u\, \mathcal{U}\, v)$	D3
	16.	$\vdash (\neg v\, \mathcal{W}\, (\neg u \wedge \neg v)) \Leftrightarrow \neg(u\, \mathcal{U}\, v)$	4, 15

viii)	1.	$\vdash u\,\mathcal{U}v \Leftrightarrow u\,\mathcal{W}v \wedge \Diamond v$	D3
	2.	$\vdash u\,\mathcal{U}v \Rightarrow u\,\mathcal{W}v$	PR
	3.	$\vdash \neg(u\,\mathcal{W}v) \Rightarrow \neg(u\,\mathcal{U}v)$	PR
	4.	$\vdash \neg(u\,\mathcal{W}v) \Rightarrow \neg v\,\mathcal{W}(\neg u \wedge \neg v)$	vi), PR
	5.	$\vdash \Box(u \vee v) \Leftrightarrow (u \vee v) \wedge \bigcirc\Box(u \vee v)$	T2i)
	6.	$\vdash \Box(u \vee v) \Rightarrow v \vee (u \wedge \bigcirc(u \vee v))$	PR
	7.	$\vdash \Box(u \vee v) \Rightarrow u\,\mathcal{W}v$	$\mathcal{W}$-I
	8.	$\vdash \neg(u\,\mathcal{W}v) \Rightarrow \Diamond(\neg u \wedge \neg v)$	PR, D1
	9.	$\vdash \neg(u\,\mathcal{W}v) \Rightarrow \neg v\,\mathcal{W}(\neg u \wedge \neg v) \wedge \Diamond(\neg u \wedge \neg v)$	4, 8, PR
	10.	$\vdash \neg(u\,\mathcal{W}v) \Rightarrow \neg v\,\mathcal{U}(\neg u \wedge \neg v)$	D3
	11.	$\vdash u \wedge \neg v \Rightarrow u \vee v$	PT
	12.	$\vdash \Box(u \wedge \neg v) \Rightarrow \Box(u \vee v)$	$\Box$-G
	13.	$\vdash ((\neg v\,\mathcal{W}(\neg u \wedge \neg v)) \wedge (u\,\mathcal{W}v) \Rightarrow \Box(u \wedge \neg v)$	vii)8
	14.	$\vdash ((\neg v\,\mathcal{W}(\neg u \wedge \neg v)) \wedge (u\,\mathcal{W}v) \Rightarrow \Box(u \vee v)$	PR
	15.	$\vdash ((\neg v\,\mathcal{W}(\neg u \wedge \neg v)) \wedge (u\,\mathcal{W}v) \Rightarrow \neg\Diamond(\neg u \wedge \neg v)$	D1, PR
	16.	$\vdash (a \wedge b \Rightarrow \neg c) \Rightarrow (a \wedge c \Rightarrow \neg b)$	PT
	17.	$\vdash (\neg v\,\mathcal{W}(\neg u \wedge \neg v)) \wedge \Diamond(\neg u \wedge \neg v) \Rightarrow \neg(u\,\mathcal{W}v)$	15, 16, PR
	18.	$\vdash (\neg v\,\mathcal{W}(\neg u \wedge \neg v)) \wedge \Diamond(\neg u \wedge \neg v) \Leftrightarrow \neg(u\,\mathcal{W}v)$	10, 17

∎

Appendix B

Bounded Past Time Temporal Logics

In chapter 5 we gave a translation of language $B\mathcal{F}\mathcal{L}$ with semantics defined in terms of infinite or bounded state sequences. In this appendix we give a similar translation [BKP84] of the language $BFPTTL$ which consists of $B\mathcal{F}\mathcal{L}$ together with past time operators. In this language, the future may be finite or infinite, but the past is always finite.

B.1 Syntax

The language BFPTTL consists of the language $B\mathcal{F}\mathcal{L}$ together with the past time operators $\bullet$, $\diamondsuit$, $\blacksquare$ and $\mathcal{S}$.

B.2 Semantics.

The models in $B\mathcal{F}\mathcal{P}\mathcal{L}$ are the those of $B\mathcal{F}\mathcal{L}$. The semantics of the additional operators is as follows

$$
\begin{array}{lll}
\langle \mathcal{M}, m_j \rangle \models \bullet f & \text{iff} & j > 0 \text{and} \langle \mathcal{M}, m_{j-1} \rangle \models f \\
\langle \mathcal{M}, m_j \rangle \models \blacksquare f & \text{iff} & \langle \mathcal{M}, m_i \rangle \models f \text{ for all } i, 0 \le i < j \\
\langle \mathcal{M}, m_j \rangle \models \diamondsuit f & \text{iff} & \langle \mathcal{M}, m_i \rangle \models f \text{ for some } i, 0 \le i < j \\
\langle \mathcal{M}, m_j \rangle \models f \mathcal{S} g & \text{iff} & \text{there is a } k, 0 \le k < j \text{ with } \langle \mathcal{M}, m_k \rangle \models g \\
 & & \text{and for all } i, k < i < j, \langle \mathcal{M}, m_i \rangle \models f.
\end{array}
$$

B.3 Translation.

Again the translation consists of a pair of maps $\widehat{\text{l}}$ and $\widehat{\text{m}}$ where

$$\tau_l{:}B\mathcal{F}\mathcal{P}\mathcal{L} \to \mathcal{F}\mathcal{P}\mathcal{L}, \tau_m{:}ModB \to Mod$$

and $ModB$, Mod denote the sets of models of $B\mathcal{F}\mathcal{L}$ and $\mathcal{F}\mathcal{L}$ respectively.

We define an operator lab in $\mathcal{FPL}$ by

$$lab(\lambda) \quad = \quad (\lambda \mathcal{S} \blacksquare \neg\lambda) \& \Box(\lambda \vee \Box\neg\lambda).$$

The map τ_l is then defined by

$$\tau_l(f) \quad = \quad lab(\lambda) \& \tau_{fin}(f, \lambda)$$

where τ_{fin} is defined as in chapter 5 for $B\mathcal{FL}$ operators, and for the past time operators by

$$\begin{array}{rcl}
\tau_{fin}(Hf, \lambda) & = & H(\Rightarrow \tau_{fin}(f, \lambda)) \\
\tau_{fin}(Pf, \lambda) & = & P(\& \tau_{fin}(f, \lambda)) \\
\tau_{fin}(Yf, \lambda) & = & Y(\& \tau_{fin}(f, \lambda)) \\
\tau_{fin}(fSg, \lambda) & = & \tau_{fin}(f, \lambda) S \tau_{fin}(g, \lambda) \\
\tau_{fin}(\neg Hf, \lambda) & = & \neg \tau_{fin}(Hf, \lambda) \\
\tau_{fin}(\neg Pf, \lambda) & = & \neg \tau_{fin}(Pf, \lambda) \\
\tau_{fin}(\neg Yf, \lambda) & = & \neg \tau_{fin}(Yf, \lambda) \\
\tau_{fin}(\neg (fSg), \lambda) & = & \neg \tau_{fin}(fSg, \lambda)
\end{array}$$

If $M = (m, v) \in ModB$ then define $\tau_m(M) = S = (s, v_1)$, where$v_1(s_i)$ is defined as in chapter 5 for$i \geq 0$, and by $v_1(s_i) = \emptyset$ for $i < 0$.

Appendix C

Sample output.

This appendix contains sample output and timings.

```
(((p & ~q ) U q ) > (p U q ))
     14 subformulae
No. of States            1
formula valid
        3.0 real         0.6 user         0.2 sys
(*----------------------------------------------------------*)
(((r + s ) U q ) > ((r U q )+ (s U q )))
     21 subformulae
No. of States           13
formula invalid
        9.0 real         4.1 user         0.4 sys
(*----------------------------------------------------------*)
((G (a A x )& G (a A y )) > G (a A (x & y )))

     39 subformulae
No. of States           37
formula valid
     2:59.0 real      1:15.7 user         1.5 sys
(*----------------------------------------------------------*)
```

(The operator `A` used in this and subsequent examples is the "latches until after" operator described in Chapter 6)

```
((G (a A (b & c ))& G (b A (a & ~c ))) > G ((a + b )A false ))
     42 subformulae
No. of States           35
formula valid
     2:09.0 real      1:23.1 user         8.8 sys
(*----------------------------------------------------------*)
```

```
(((a A b )& (b A c )) > ((a + b )A c ))
     34 subformulae
No. of States            40
formula valid
     1:16.0 real        39.6 user         2.5 sys
(*------------------------------------------------------------*)
(H p > p )
      7 subformulae
No. of States             2
formula invalid
        2.0 real         0.3 user         0.3 sys
(*------------------------------------------------------------*)
(F P p > (F p + P p ))
     15 subformulae
No. of States            17
formula valid
        7.0 real         5.8 user         0.5 sys
(*------------------------------------------------------------*)
((P p & G H (p > X p )) > G p )
     17 subformulae
No. of States            39
formula valid
     1:13.0 real      1:01.9 user         3.0 sys
(*------------------------------------------------------------*)
```

The following two examples show the graphs produced by the algorithm.

```
(((r + s ) U q ) > ((r U q )+ (s U q )))

(((r + s ) U q )& ((~q W (~r & ~q ))& (~q W (~s & ~q ))))

Form key complete     21 subformulae
          Formula Key

formula no.    0  false
        Atom
formula no.    1  true
        Atom
formula no.    2  (((r + s ) U q )& ((~q W (~r & ~q ))&
                              (~q W (~s & ~q ))))
        alpha    3   4
formula no.    3  ((r + s ) U q )
        beta   5   6  Eventy    5
formula no.    4  ((~q W (~r & ~q ))& (~q W (~s & ~q )))
```

```
        alpha    7   8
formula no.    5  q
        Atom
formula no.    6  ((r + s )& X ((r + s ) U q ))
        alpha    9  10
formula no.    7  (~q W (~r & ~q ))
        beta  11  12
formula no.    8  (~q W (~s & ~q ))
        beta  13  14
formula no.    9  (r + s )
        beta  15  16
formula no.   10  X ((r + s ) U q )
        Next    3
formula no.   11  (~r & ~q )
        alpha   17  18
formula no.   12  (~q & X (~q W (~r & ~q )))
        alpha   18  19
formula no.   13  (~s & ~q )
        alpha   20  18
formula no.   14  (~q & X (~q W (~s & ~q )))
        alpha   18  21
formula no.   15  r
        Atom
formula no.   16  s
        Atom
formula no.   17  ~r
        Not   15
formula no.   18  ~q
        Not    5
formula no.   19  X (~q W (~r & ~q ))
        Next    7
formula no.   20  ~s
        Not   16
formula no.   21  X (~q W (~s & ~q ))
        Next    8

No. of States             13

          ***PreStates***


PreState
  Label        3
States  11  12  13
```

```
PreState
  Label        3   7
States   8   9  10

PreState
  Label        3   7   8
States   1   2   3   4

PreState
  Label        3   8
States   5   6   7

          ***States***
State no.          1
  Label        3   7   8   9  10  16  18  19  21
  Nexts        3   7   8
  Lasts
Successors   1   2   3   4
Predecessors   1   2

State no.          2
  Label        3   7   8   9  10  15  18  19  21
  Nexts        3   7   8
  Lasts
Successors   1   2   3   4
Predecessors   1   2

State no.          3
  Label        3   7   8   9  10  16  17  18  21
  Nexts        3   8
  Lasts
Successors   5   6   7
Predecessors   1   2

State no.          4
  Label        3   7   8   9  10  15  18  19  20
  Nexts        3   7
  Lasts
Successors   8   9  10
Predecessors   1   2

State no.          5
  Label        3   8   9  10  16  18  21
  Nexts        3   8
  Lasts
```

```
Successors   5   6   7
Predecessors   3   5   6

State no.          6
  Label        3   8   9  10  15  18  21
  Nexts        3   8
  Lasts
Successors   5   6   7
Predecessors   3   5   6

State no.          7
  Label        3   8   9  10  15  18  20
  Nexts        3
  Lasts
Successors  11  12  13
Predecessors   3   5   6

State no.          8
  Label        3   7   9  10  16  18  19
  Nexts        3   7
  Lasts
Successors   8   9  10
Predecessors   4   8   9

State no.          9
  Label        3   7   9  10  15  18  19
  Nexts        3   7
  Lasts
Successors   8   9  10
Predecessors   4   8   9

State no.         10
  Label        3   7   9  10  16  17  18
  Nexts        3
  Lasts
Successors  11  12  13
Predecessors   4   8   9

State no.         11
  Label        3   9  10  16
  Nexts        3
  Lasts
Successors  11  12  13
Predecessors   7  10  11  12
```

```
State no.          12
  Label        3   9  10  15
  Nexts        3
  Lasts
Successors  11  12  13
Predecessors   7  10  11  12

State no.          13
  Label        3   5
  Nexts
  Lasts
Successors
Predecessors   7  10  11  12



(*-----------------------------------------------------------*)
        Formula Invalid
(*-----------------------------------------------------------*)

          *** Final State Graph ***

State no.           1
  Label        3   7   8   9  10  16  18  19  21
  Nexts        3   7   8
  Lasts
Successors   1   2   3   4
Predecessors   1   2

State no.           2
  Label        3   7   8   9  10  15  18  19  21
  Nexts        3   7   8
  Lasts
Successors   1   2   3   4
Predecessors   1   2

State no.           3
  Label        3   7   8   9  10  16  17  18  21
  Nexts        3   8
  Lasts
Successors   5   6   7
Predecessors   1   2

State no.           4
  Label        3   7   8   9  10  15  18  19  20
```

```
  Nexts        3   7
  Lasts
Successors   8   9  10
Predecessors   1   2

State no.          5
  Label        3   8   9  10  16  18  21
  Nexts        3   8
  Lasts
Successors   5   6   7
Predecessors   3   5   6

State no.          6
  Label        3   8   9  10  15  18  21
  Nexts        3   8
  Lasts
Successors   5   6   7
Predecessors   3   5   6

State no.          7
  Label        3   8   9  10  15  18  20
  Nexts        3
  Lasts
Successors  11  12  13
Predecessors   3   5   6

State no.          8
  Label        3   7   9  10  16  18  19
  Nexts        3   7
  Lasts
Successors   8   9  10
Predecessors   4   8   9

State no.          9
  Label        3   7   9  10  15  18  19
  Nexts        3   7
  Lasts
Successors   8   9  10
Predecessors   4   8   9

State no.         10
  Label        3   7   9  10  16  17  18
  Nexts        3
  Lasts
Successors  11  12  13
```

```
Predecessors   4   8   9

State no.          11
  Label        3   9  10  16
  Nexts        3
  Lasts
Successors  11  12  13
Predecessors   7  10  11  12

State no.          12
  Label        3   9  10  15
  Nexts        3
  Lasts
Successors  11  12  13
Predecessors   7  10  11  12

State no.          13
  Label        3   5
  Nexts
  Lasts
Successors
Predecessors   7  10  11  12

       10.0 real         7.0 user         0.5 sys
(*-----------------------------------------------------------*)
(F P p > (F p + P p ))

(F P p & (G ~p & H ~p ))

Form key complete     15 subformulae
          Formula Key

formula no.    0  false
        Atom
formula no.    1  true
        Atom
formula no.    2  (F P p & (G ~p & H ~p ))
        alpha    3   4
formula no.    3  F P p
        beta   5   6  Eventy    5
formula no.    4  (G ~p & H ~p )
        alpha    7   8
formula no.    5  P p
        beta   9  10  Eventy   -9
formula no.    6  X F P p
```

```
        Next    3
formula no.    7  G ~p
        alpha   11  12
formula no.    8  H ~p
        alpha   13  14
formula no.    9  Y p
        Last   15
formula no.   10  Y P p
        Last    5
formula no.   11  ~p
        Not   15
formula no.   12  X G ~p
        Next    7
formula no.   13  Y ~p
        Last   11
formula no.   14  Y H ~p
        Last    8
formula no.   15  p
        Atom

No. of States             17

          ***PreStates***

PreState
  Label        5
States  15  16

PreState
  Label        3  11  12
States   5   6   7

PreState
  Label        3   5   6  11  12
States  13  14

PreState
  Label        3  11  12  13  14
States   1   2   3

PreState
  Label        3   5   6  11  12  13  14
States   8   9

PreState
```

```
  Label        5  11  13  14
States  10  11

PreState
  Label       11  12
States  12

PreState
  Label       11  13  14
States   4

PreState
  Label       15
States  17

          ***States***

State no.          1
  Label        3   6  11  12  13  14
  Nexts        3  11  12
  Lasts       11  13  14
Successors   5
Predecessors   4
Augmented States   8   9

State no.          2
  Label        3   5  10  11  12  13  14
  Nexts       11  12
  Lasts        5  11  13  14
Successors  12
Predecessors  10  11
Augmented States

State no.          3
  Label        3   5   9  11  12  13  14
  Nexts       11  12
  Lasts       11  13  14  15
Successors  12
Predecessors
Augmented States

State no.          4
  Label       11  13  14
  Nexts
  Lasts       11  13  14
```

```
Successors   1   4
Predecessors   4
Augmented States

State no.          5
  Label        3   6  11  12
  Nexts        3  11  12
  Lasts
Successors   5
Predecessors   1   5   8   9  13  14
Augmented States  13  14

State no.          6
  Label        3   5  10  11  12
  Nexts       11  12
  Lasts        5
Successors  12
Predecessors   8   9  13  14  15  16
Augmented States

State no.          7
  Label        3   5   9  11  12
  Nexts       11  12
  Lasts       15
Successors  12
Predecessors  17
Augmented States

State no.          8
  Label        3   5   6  10  11  12  13  14
  Nexts        3  11  12
  Lasts        5  11  13  14
Successors   5   6  13
Predecessors  10  11
Augmented States

State no.          9
  Label        3   5   6   9  11  12  13  14
  Nexts        3  11  12
  Lasts       11  13  14  15
Successors   5   6  13
Predecessors
Augmented States

State no.         10
```

```
  Label        5  10  11  13  14
  Nexts
  Lasts        5  11  13  14
Successors   2   8  10
Predecessors  10  11
Augmented States

State no.          11
  Label        5   9  11  13  14
  Nexts
  Lasts       11  13  14  15
Successors   2   8  10
Predecessors
Augmented States

State no.          12
  Label       11  12
  Nexts       11  12
  Lasts
Successors  12
Predecessors   2   3   6   7  12
Augmented States

State no.          13
  Label        3   5   6  10  11  12
  Nexts        3  11  12
  Lasts        5
Successors   5   6  13
Predecessors   8   9  13  14  15  16
Augmented States

State no.          14
  Label        3   5   6   9  11  12
  Nexts        3  11  12
  Lasts       15
Successors   5   6  13
Predecessors  17
Augmented States

State no.          15
  Label        5  10
  Nexts
  Lasts        5
Successors   6  13  15
Predecessors  15  16
```

```
Augmented States

State no.          16
  Label        5   9
  Nexts
  Lasts       15
Successors   6  13  15
Predecessors  17
Augmented States

State no.          17
  Label       15
  Nexts
  Lasts
Successors   7  14  16
Predecessors
Augmented States


(*-----------------------------------------------------------*)
   Formula Valid
(*-----------------------------------------------------------*)
          *** Final State Graph ***

State no.           4
  Label       11  13  14
  Nexts
  Lasts       11  13  14
Successors   4
Predecessors   4
Augmented States

State no.           6
  Label        3   5  10  11  12
  Nexts       11  12
  Lasts        5
Successors  12
Predecessors  13  14  15  16
Augmented States

State no.           7
  Label        3   5   9  11  12
  Nexts       11  12
  Lasts       15
Successors  12
```

```
Predecessors  17
Augmented States

State no.          12
  Label       11  12
  Nexts       11  12
  Lasts
Successors  12
Predecessors   6   7  12
Augmented States

State no.          13
  Label        3   5   6  10  11  12
  Nexts        3  11  12
  Lasts        5
Successors   6  13
Predecessors  13  14  15  16
Augmented States

State no.          14
  Label        3   5   6   9  11  12
  Nexts        3  11  12
  Lasts       15
Successors   6  13
Predecessors  17
Augmented States

State no.          15
  Label        5  10
  Nexts
  Lasts        5
Successors   6  13  15
Predecessors  15  16
Augmented States

State no.          16
  Label        5   9
  Nexts
  Lasts       15
Successors   6  13  15
Predecessors  17
Augmented States

State no.          17
  Label       15
```

```
  Nexts
  Lasts
Successors   7  14  16
Predecessors
Augmented States

       19.0 real         9.1 user         0.6 sys
(*-------------------------------------------------------------*)
```

References

[BAMP81] M. Ben-Ari, Z. Manna, and A. Pnueli.
The Temporal Logic of Branching Time.
Proceedings of the Eighth ACM Symposium on the Principles of Programming Languages, pages 164–176, 1981.

[BK83] H. Barringer and R. Kuiper.
Towards the Hierarchical, Temporal Logic Specification of Concurrent Systems.
Lecture Notes in Computer Science, 207, September 1983.

[BKP84] H. Barringer, R. Kuiper, and A. Pnueli.
Now You May Compose Temporal Logic Specifications.
In *Proceedings of the Sixteenth ACM Symposium on the Theory of Computing*, 1984.

[CES86] E. M. Clarke, E. A. Emerson, and A. P. Sistla.
Automatic verification of finite-state concurrent systems using temporal logic specifications.
ACM Transactions on Programming Languages and Systems, 8(2):244–263, 1986.

[EH82] E. A. Emerson and J. Y. Halpern.
Decision procedures and expressiveness in the temporal logic of branching time.
Proceedings of the Fourteenth ACM Symposium on the Theory of Computing, 1982.

[FL79] M. J. Fischer and R. L. Ladner.
Propositional dynamic logic of regular programs.
Journal of Computer and System Sciences, 18:194–211, 1979.

[Gri71] D. Gries.
Compiler construction for Digital Computers.
Wiley, 1971.

[HC72] G. E. Hughes and M. J. Cresswell.
An Introduction to Modal Logic., volume 431 of *University Paperbacks*.
Methuen, 1972.

[MP82] Z. Manna and A. Pnueli.
Verification of Concurrent Programs: The Temporal Framework.
In Robert S. Boyer and J. Strother Moore, editors, *The Correctness Problem in Computer Science*. Academic Press, London, 1982.

[MP83] Z. Manna and A. Pnueli.
Verification of Concurrent Programs: A Temporal Proof System.
Technical report, Department of Computer Science, Stanford University, 1983.

[MW84] Z. Manna and P. Wolper.
Synthesis of communicating processes from temporal logic specifications.
ACM Transactions on Programming Languages and Systems, 6(1), January 1984.

[Pla83a] D. A. Plaisted.

A decision procedure for combinations of propositional temporal logic and other specialized theories.
Technical Report 172262, SRI International, 1983.

[Pla83b] D. A. Plaisted.
A low level language for obtaining decision procedures for classes of temporal logics.
Technical Report 172262, SRI International, 1983.

[Pnu77] A. Pnueli.
The Temporal Logic of Programs.
In *Proceedings of the Eighteenth Symposium on the Foundations of Computer Science*, Providence, November 1977.

[RU71] N. Rescher and A. Urquart.
Temporal logic.
Springer-Verlag, 1971.

[SMSVP83] R. L. Schwartz, P. M. Melliar-Smith, F. H. Vogt, and D. A. Plaisted.
An interval logic for higher-level temporal reasoning.
Technical Report 172262, SRI International, September 1983.

[Smu68] R. M. Smullyan.
First Order Logic, volume 43 of *Ergebnisse der Mathematik*.
Springer-Verlag, 1968.

[Wol83] P. Wolper.
Temporal logic can be more expressive.
Information and Control, 56, 1983.