Acta Informatica 20, 207–226 (1983)

Acta Informatica

© Springer-Verlag 1983

The Temporal Logic of Branching Time* **

Mordechai Ben-Ari[1], Amir Pnueli[2], and Zohar Manna[2,3]

[1] Department of Computer Science, School of Mathematical Sciences, Tel Aviv University, Ramat Aviv 69978, Tel Aviv, Israel
[2,3] Department of Applied Mathematics, The Weizmann Institute of Science, Rehovot, Israel 76100
[3] Department of Computer Science, Stanford University, Stanford, CA 94305, USA

Summary. A temporal logic is defined which contains both linear and branching operators. The underlying model is the tree of all possible computations. The following metatheoretical results are proven: 1) an exponential decision procedure for satisfiability; 2) a finite model property; 3) the completeness of an axiomatization.

1. Introduction

From the first introduction of the Temporal Logic formalism as a tool for reasoning about programs, there arose a basic question which later almost developed into a controversy. The question involves the nature of the underlying structure of time on which the formalism is based. The dichotomy is between the linear time approach which considers time to be a linear sequence, and the branching time approach, which adopts a tree structured time, allowing some instants to have more than a single successor.

The difference in approaches has very little to do with the philosophical question of the structure of physical time which leads to the metaphysical problems of determinancy versus free will.

Instead it is pragmatically based on the choice of the type of programs and properties one wishes to formalize and study.

Linear time is the correct model to use in order to characterize the set of all execution sequences which a program generates and to study properties which uniformly hold for all the execution sequences of a program. Even a

* Portions of this work are based on the first author's Ph.D. thesis done at the Tel Aviv University under the supervision of the second author. A Preliminary version of this paper was presented at the Eighth ACM Symposium on Principles of Programming Languages, Williamsburg, VA, 1981

** This research was supported in part by: NSF under grant MCS-80-6930, Office of Naval Research under grant N000-14-76-C-0687, the United States Air Force Office of Scientific Research under Grant AFOSR-81-0014, and the Israeli Academy of Sciences and Humanities, the Basic Research Foundation

nondeterministic program generates for each set of possible choices a linear execution sequence in which each execution state has a unique successor. The class of properties related to the set of execution sequences are the *universal* properties such as universal total correctness and universal responsiveness. Both these properties require that *every* execution sequence of the program will eventually achieve some goal such as termination with a correct result or correct response to some request. The interpretation of temporal formulas over execution sequences of a given program was found to be very useful for reasoning about both sequential deterministic programs and concurrent programs. In the case of concurrent programs, where the nondeterminism is caused by different scheduling scripts, we generally wish to prove that the program terminates or responds correctly regardless of how the individual processes are scheduled. This approach is pursued in [9, 11, 12].

The branching time approach, on the other hand, considers for a given program the set of all *execution trees* generated by the program. With a nondeterministic program P and a given input x we can associate the tree of all possible computations of P on x. Since the program is nondeterministic, some of the execution states will have more than one successor corresponding to a nondeterministic choice. Over execution trees we can study *existential* properties such as correct termination for at least one possible computation (for every input). More generally, we may study the property that there is always one possible computation which realizes some goal. This certainly does not imply that all computations will realize the same goal. Consequently, this approach is useful for nondeterministic programs which are executed by systematically exploring all possible choices by methods such as breadth first search, etc. This interpretation of nondeterminism is recommended for example in [4] as a design tool and is the one classically used in automata and complexity theory. The branching time approach is implied in the underlying structure of Dynamic Logic ([1, 6]), but was not previously studied in a temporal framework.

In the end, the choice between linear and branching models cannot be made on philosophical grounds but instead should be dictated by the type of programs, execution policies and properties which one wishes to study. For a fuller discussion of this issue see [9].

A natural step at this point would be to formalize and investigate a branching time temporal logic which incorporates both universal operators similar to those used in linear temporal logic, and existential operators similar to those used in simpler branching systems. It turns out that such a unified system which combines both approaches is not significantly more complex than the two separate systems.

We will define a logic $\mathcal{UB}$: the *unified* system of *branching* time. The underlying model will be the branching tree of all possible computations of a program. We define, however, additional temporal operators that allow reference either to all possible execution sequences or only to a single sequence. The metatheoretical results in $\mathcal{UB}$ include:

1) An exponential decision procedure for satisfiability.

2) A finite model property: If a formula in $\mathcal{UB}$ is satisfiable then it has a finite (exponential) model.

3) A simple axiomatization which is shown to be complete.

The decision procedure uses semantic tableaux. Tableau systems provide a rapid way of deciding "natural" formulas. The completeness theorem shows how to "read-off" a proof from a tableau.

The expressive power of the system is illustrated by formalizing both universal and existential properties of nondeterministic programs. Finally we give a temporal semantics for nondeterministic programs, complementing the semantics given in [12] for deterministic but concurrent programs. Even though the $\mathcal{UB}$ logic, inspired by linear temporal logic, incorporates universal operators that correspond to statements holding in all paths of the model tree, it cannot be regarded as a generalization of linear temporal logic. In fact the two logics are incomparable in the sense that there are statements expressible in each which are not expressible in the other. Thus, even though the $\mathcal{UB}$ logic significantly extends the branching time logic studied in [9], the same essential result of [9], i.e. incomparability of linear versus branching time logic still holds. The examples for incomparability are essentially the same examples presented in [9].

The statement that there exists some convergent computation of a given program is expressible in $\mathcal{UB}$ but not in linear temporal logic. On the other hand the statement that all fair computations of a concurrent program converge is expressible in linear time temporal logic but not in $\mathcal{UB}$.

In Sect. 2 the syntax and semantics of the system $\mathcal{UB}$ (unified branching) are given along with the axiom system. Section 3 contains a list of theorems of $\mathcal{UB}$ along with a representative selection of proofs. Section 4 describes the construction of semantic tableaux for $\mathcal{UB}$ giving a decision procedure and the finite model property. Section 5 proves completeness of the axiom system from the tableaux of Sect. 4. In Sect. 6 we illustrate the utility of the $\mathcal{UB}$ system for axiomatizing the behavior of nondeterministic programs. Such axiomatization would lead to proof methods for proving properties of nondeterminstic programs.

An earlier abstract of this work appeared in [2]. This earlier version contained an error which is corrected in the current paper.

2. The System $\mathcal{UB}$

The base of $\mathcal{UB}$ is the propositional calculus on $\sim$ and $\vee$ with the other connectives defined as usual.

We use two symbols for each modality. The first, $\forall$ or $\exists$, denotes quantification over branches. The second G, F or X (*nexttime*) denotes quantification over states in the branches. The three primitive modalities are $\forall G$, $\exists G$ and $\forall X$. The three defined dual modalities are $\exists F = \sim\forall G\sim$; $\forall F = \sim\exists G\sim$; $\exists X = \sim\forall X\sim$. Parentheses may be omitted by defining $\sim$ and the modalities to have precedence over $\wedge$ which in turn has precedence over $\vee$ and $\supset$.

Let T be a tree and s a node in T. Let a be a proposition which can hold at some nodes in T. The intuitive meaning of the modal operators when applied to a proposition is as follows.

$\forall G a$ holds at s (in T) iff a is true at all nodes of the subtree rooted at s (including s).

$\forall F a$ holds at s iff on every path departing from s there is some node at which a is true.

$\forall X a$ holds at s iff a is true at every immediate successor of s.

$\exists G a$ holds at s iff there is a path departing from s such that a is true at all nodes on this path.

$\exists F a$ holds at s iff a is true at some node in the subtree rooted at s, i.e. there is a path departing from s such that a is true at some node on this path.

$\exists X a$ holds at s iff a is true at one of the immediate successors of s.

The four modalities $\forall G$, $\forall F$, $\exists G$, $\exists F$ correspond to four concepts of correctness given by Manna [10] for a general nondeterministic program P. Let B be the proposition that a program P is at its start state and H be the proposition that P has halted. Let φ and Ψ be the input and output predicates which form a correctness specification for P. Then Manna distinguishes four concepts of correctness of P relative to (φ, Ψ).

a) $(B \wedge \varphi) \supset \exists G(H \supset \Psi)$.

P is *partially* $\exists$*-correct* with respect to (φ, Ψ).

If the program is started with a true input specification then there exists a computation that may be infinite, but if it is finite then the program terminates in a state satisfying the output specification. (Actually, the program halts by remaining forever in the halt state to conform to our model of non-ending time.)

b) $(B \wedge \varphi) \supset \exists F(H \wedge \Psi)$.

P is *totally* $\exists$*-correct* with respect to (φ, Ψ).

There is a computation that terminates correctly, though other computations may diverge or terminate incorrectly.

c) $(B \wedge \varphi) \supset \forall G(H \supset \Psi)$.

P is *partially* $\forall$*-correct* with respect to (φ, Ψ).

Every terminating computation must be correct but there is no guarantee that any terminating computation exists.

d) $(B \wedge \varphi) \supset \forall F(H \wedge \Psi)$.

P is *totally* $\forall$*-correct* with respect to (φ, Ψ).

Every computation terminates correctly.

Semantics for $\mathcal{UB}$

A model for $\mathcal{UB}$ is a triple (S, P, R) where S is a set of states, P is an assignment of propositional letters to states and R is a binary relation on states. If sRt then we say that t is an immediate successor of s. To capture the concept of non-ending time, we require that R be total, i.e. $\forall s \exists t\ (sRt)$.

For a propositional letter a and a state $s \in S$, $a \in P(s)$ iff a is true in s. We extend the interpretation over the model to all formulas in $\mathcal{UB}$ as follows where $b = (s = s_0, s_1, s_2, \ldots)$ is an infinite path through the model such that $s_i R s_{i+1}$. b is called an s-branch.

1. $s \models a$ iff $a \in P(s)$, for a atomic
2. $s \models \sim p$ iff $s \not\models p$
3. $s \models p \vee q$ iff $s \models p$ or $s \models q$
4. $s \models \forall G p$ iff $\forall b \forall t$ ($t \in b$ implies $t \models p$)
5. $s \models \exists G p$ iff $\exists b \forall t$ ($t \in b$ implies $t \models p$)
6. $s \models \forall X p$ iff $\forall t$ ($s R t$ implies $t \models p$)
7. $s \models \exists F p$ iff $\exists b \exists t$ ($t \in b$ and $t \models p$)
8. $s \models \forall F p$ iff $\forall b \exists t$ ($t \in b$ and $t \models p$)
9. $s \models \exists X p$ iff $\exists t$ ($s R t$ and $t \models p$).

p is *satisfiable* if $s \models p$ for some model M and some state s in M; this is written $M, s \models p$, or $s \models p$ if M is understood. In 7, 8 and 9 we say the formula in s has been fulfilled at t.

The following is a set of axioms for $\mathscr{UB}$.

A 1. $\forall G(p \supset q) \supset (\forall G p \supset \forall G q)$
A 2. $\forall X(p \supset q) \supset (\forall X p \supset \forall X q)$
A 3. $\forall G p \supset \forall X p \wedge \forall X \forall G p$
A 4. $\forall G(p \supset \forall X p) \supset (p \supset \forall G p)$
E 1. $\forall G(p \supset q) \supset (\exists G p \supset \exists G q)$
E 2. $\exists G p \supset p \wedge \exists X \exists G p$
E 3. $\forall G p \supset \exists G p$
E 4. $\forall G(p \supset \exists X p) \supset (p \supset \exists G p)$.

The rules of inference are:

R 1. If p is a substitution instance of a tautology then $\vdash p$.
R 2. If $\vdash p$ and $\vdash p \supset q$ then $\vdash q$. (Modus ponens)
R 3. If $\vdash p$ then $\vdash \forall G p$. (Necessitation).

If $\forall G p \supset p$ (T 1, below) is added to A 1–A 4, then we get a complete axiomatization of the universal fragment of branching nexttime. If $\forall X$ and $\exists X$ are merged (along with $\forall G$ and $\exists G$) so that $\forall X p \equiv \sim \forall X \sim p$ is an axiom then we get a complete axiomatization of linear nexttime. By T 1 and T 5, below, we could express A 3 and E 2 more symmetrically as:

A 3′. $\forall G p \supset p \wedge \forall X p \wedge \forall X \forall G p$
E 2′. $\exists G p \supset p \wedge \exists X p \wedge \exists X \exists G p$.

Also, E 3 can be derived by taking T 1 and T 6 as axioms. Some axiom of the form $\forall \supset \exists$ is needed to limit our models to non-ending time.

3. Theorems of $\mathscr{UB}$

T 1. $\forall G p \supset p$
T 2. $\forall G p \supset \forall F p$
T 3. $\forall X(p \supset q) \supset (\exists X p \supset \exists X q)$
T 4. $\forall G(p \supset q) \supset (\forall F p \supset \forall F q)$
T 5. $\exists G p \supset \exists X p$
T 6. $\forall X p \supset \exists X p$

T7. $\forall G(p \wedge q) \equiv \forall G p \wedge \forall G q$
T8. $\exists G(p \wedge q) \supset \exists G p \wedge \exists G q$
T9. $\forall X(p \wedge q) \equiv \forall X p \wedge \forall X q$
T10. $\exists X(p \wedge q) \supset \exists X p \wedge \exists X q$
T11. $\forall X p \wedge \exists X q \supset \exists X(p \wedge q)$
T12. $\forall G p \wedge \exists G q \supset \exists G(p \wedge q)$
T13. $\forall G p \equiv p \wedge \forall X \forall G p$
T14. $\exists G p \equiv p \wedge \exists X \exists G p$
T15. $\forall G p \equiv \forall G \forall G p$
T16. $\exists G p \equiv \exists G \exists G p$
T17. $\exists G(p \supset \forall X p) \supset (p \supset \exists G p)$
T18. $\forall F \forall G p \supset \forall G \forall F p$
T19. $\exists G((p \vee \exists G q) \wedge (\exists G p \vee q)) \equiv (\exists G p \vee \exists G q)$
T20. $\forall X \forall G p \equiv \forall G \forall X p$
T21. $\exists X \exists G p \supset \exists G \exists X p$.

From A1-A3, E1, E3, T1-T6 we can derive necessitation rules $\vdash p \rightarrow \vdash Mp$ and $\vdash p \supset q \rightarrow \vdash Mp \supset Mq$ for all modalities $M \in \{\forall G, \exists G, \forall F, \exists F, \forall X, \exists X\}$. These derived rules are used implicitly below.

The proofs of T1-T12 are straightforward. T9 and T11 form the key to inductive proofs as will be seen in the completeness proof. The rest of the theorems are proven using the induction axioms A4 and E4.

T14.
1. $\exists G p \supset p \wedge \exists X \exists G p$ E2
2. $\exists X \exists G p \supset \exists X(p \wedge \exists X \exists G p)$
3. $p \wedge \exists X \exists G p \supset \exists X(p \wedge \exists X \exists G p)$
4. $\forall G(p \wedge \exists X \exists G p \supset \exists X(p \wedge \exists X \exists G p))$
5. $p \wedge \exists X \exists G p \supset \exists G(p \wedge \exists X \exists G p)$ E4
6. $p \wedge \exists X \exists G p \supset \exists G p$ T8
7. $\exists G p \equiv p \wedge \exists X \exists G p$ 1, 6

In their negated forms T13 and T14 are the key to the tableau constructions used in the meta-theory of $\mathcal{UB}$: $\sim \exists G p \supset \sim p \vee \sim \exists X \exists G p$. If we try to falsify $\exists G p$ then either falsify it now ($\sim p$) or if that is not possible, put it off to tomorrow ($\sim \exists X \exists G p \equiv \forall X \sim \exists G p$).

T15-T16 are the S4 transitivity axioms.

T17 is another induction theorem. We conjecture that replacing E4 by T17 results in a weaker system for proof theoretical reasons because the induction step $p \supset \forall X p$ is usually not true. Probably the system is not different from linear nexttime.

T18 is our version of the S4.2 axiom $\Diamond \Box p \supset \Box \Diamond p$.

Note that $\exists G p \supset \forall F p$ and $\forall F \exists G p \supset \exists G \forall F p$ can be proved but this is an artifact of the reflexiveness of $\mathcal{UB}$ and would not carry over if E2 were changed to $\exists G p \supset EX p \wedge \exists X \exists G p$ to obtain an irreflexive system as required classically in temporal logic.

T19 is the S4.3 linearity axiom for $\exists G$.

T19. Denote $(p \vee \exists G q) \wedge (\exists G p \vee q)$ by r and $\exists G r \wedge \sim \exists G p \wedge \sim \exists G q$ by s.
1. $s \supset (p \vee \exists G q) \wedge (\exists G p \vee q) \wedge \sim \exists G p \wedge \sim \exists G q$ E2

2. $s \supset p \wedge q$
3. $s \supset p \wedge q \wedge (\sim p \vee \sim \exists X \exists G p) \wedge (\sim q \vee \sim \exists X \exists G q)$ T 14
4. $s \supset \sim \exists X \exists G p \wedge \sim \exists X \exists G q$
5. $s \supset \exists X \exists G r \wedge \sim \exists X \exists G p \wedge \sim \exists X \exists G q$ E 2
6. $s \supset \exists X (\exists G r \wedge \sim \exists G p \wedge \sim \exists G q)$ T 11
   i.e., $s \supset \exists X s$
7. $\exists G (s \supset \exists X s)$
8. $s \supset \exists G s$ E 4
9. $\exists G s \supset \exists G (p \wedge q)$ 2, necessitation
10. $s \supset \exists G p \wedge \exists G q$ 8, 9, T 8
    i.e., $\exists G r \wedge \sim \exists G p \wedge \sim \exists G q \supset \exists G p \wedge \exists G q$
11. $\exists G r \supset \exists G p \vee \exists G q$
    i.e., $\exists G ((p \vee \exists G q) \wedge (\exists G p \vee q)) \supset \exists G p \vee \exists G q$.

Conversely,

12. $\exists G p \supset \exists G p \vee q$
13. $\exists G p \supset p \vee \exists G q$ E 2
14. $\exists G p \supset (\exists G p \vee q) \wedge (p \vee \exists G q)$ denote the consequent by r.
15. $\exists G \exists G p \supset \exists G r$
16. $\exists G p \supset \exists G r$ T 16
17. $\exists G q \supset \exists G r$ Symmetry
18. $\exists G p \vee \exists G q \supset \exists G r$
19. $\exists G r \equiv \exists G p \vee \exists G q$ 11, 18.

Why does the proof not work for $\forall G$? In 5 we would have:

5′. $s \supset \forall X \forall G r \wedge \sim \forall X \forall G p \wedge \sim \forall X \forall G q$
i.e. $s \supset \forall X \forall G r \wedge \exists X \sim \forall G p \wedge \exists X \sim \forall G q$

which cannot have a modality extracted.

T20. 1. $\forall G \forall G p \supset \forall G \forall X p$ A 3
2. $\forall G p \supset \forall G \forall X p$ T 15
3. $\forall X \forall G p \supset \forall X \forall G \forall X p$
4. $\forall X \forall G p \supset \forall X p$ T 1
5. $\forall X \forall G p \supset \forall X p \wedge \forall X \forall G \forall X p$ 3, 4
6. $\forall X \forall G p \supset \forall G \forall X p$ T 13
7. $\forall G \forall X p \supset \forall X p \wedge \forall X \forall G \forall X p$ T 13
8. $p \wedge \forall G \forall X p \supset \forall X (p \wedge \forall G \forall X p)$ T 9
9. $p \wedge \forall G \forall X p \supset \forall G p$ A 4, T 7
10. $\forall X p \wedge \forall X \forall G \forall X p \supset \forall X \forall G p$ A 2, T 7
11. $\forall G \forall X p \supset \forall X \forall G p$ T 13
12. $\forall X \forall G p \equiv \forall G \forall X p$ 6, 11

4. Semantic Tableaux for $\mathcal{UB}$

In this section we describe the construction of semantic tableaux for formulas in $\mathcal{UB}$, obtain a decision procedure for satisfiability and prove the finite model

property. In the next section we prove the completeness of the axiomatization of $\mathcal{UB}$ by showing that if the tableau construction fails to produce a model for $\sim p$ then we can read off the tableau a proof of p.

A *structure* is a triple (S, P, R) where S is a set of states, P is an assignment of formulas to states and R is a binary relation on states. It is convenient to use the same name S for a structure and its set of states. A structure differs from a model in that it is not required to be complete, i.e. a state need not contain p or $\sim p$ for every p. Also, we have not required any connection between the assignment of formulas and the assignment of its subformulas. We do require that our structure be appropriate for $\mathcal{UB}$, i.e. for every $s \in S$ there is a t such that sRt.

Notation. $p \in s$ is short for $p \in P(s)$.

A formula is *elementary* if it is a proposition, a negation of a proposition or a nexttime formula. Following Pratt [13], it is convenient to regard reduction of a double negation and replacement of a modality by its dual as part of the syntax of the language. Thus compound formulas are always non-negated. Obviously, this does not affect satisfiability or provability.

Following Smullyan [16] we classify formulas as α-formulas and β-formulas.

α	α_1	α_2
$p \wedge q$	p	q
$\forall G p$	p	$\forall X \forall G p$
$\exists G p$	p	$\exists X \exists G p$
β	β_1	β_2
$p \vee q$	p	q
$\forall F p$	p	$\forall X \forall F p$
$\exists F p$	p	$\exists X \exists F p$

A structure H is called a *Hintikka structure* iff for all $s \in H$:

$H1$. $\sim p \in s \Rightarrow p \notin s$ (H is consistent).
$H2$. $\alpha \in s \Rightarrow \alpha_1 \in s$ and $\alpha_2 \in s$.
$H3$. $\beta \in s \Rightarrow \beta_1 \in s$ or $\beta_2 \in s$.
$H4a$. $\forall X p \in s \Rightarrow$ for all t such that sRt, $p \in t$.
$H4b$. $\exists X p \in s \Rightarrow$ for some t such that sRt, $p \in t$.
$H4c$. $\exists F p \in s \Rightarrow$ there exists an s-branch b and for some $t \in b$, $p \in t$.
$H4d$. $\forall F p \in s \Rightarrow$ for all s-branches b there exists a $t \in b$ such that $p \in t$.

H is called a Hintikka structure *for* p if $p \in s$ for some $s \in H$.

Theorem 4.1. (Hintikka's Lemma for $\mathcal{UB}$): *A $\mathcal{UB}$ formula p_0 is satisfiable iff there is a Hintikka structure for p_0.*

Proof. ($\Rightarrow$): It is easy to check that the "only-if" directions of the model conditions 1.-9. are sufficient to guarantee that any model can be viewed as Hintikka structure.

($\Leftarrow$): Let $H = (S, P, R)$ be a Hintikka structure for p_0.

Define $H'=(S,P,R)$ by:

$$a\in P'(s) \quad \text{if} \quad a\in P(s) \text{ or } \sim a\notin P(s)$$
$$a\notin P'(s) \quad \text{if} \quad \sim a\in P(s); \text{ for atomic formulas } a.$$

Then H' defines a $\mathcal{UB}$ model. We have to show that H' is a $\mathcal{UB}$-model for p_0. This is done by a simultaneous induction showing

(i) $p\in s\in H\Rightarrow H',\ s\models p$

(ii) $\sim p\in s\in H\Rightarrow H',\ s\models \sim p$

for positive formulas p.

Basis: $a\in s\in H\Rightarrow H',\ s\models a$ by construction.
$\sim a\in s\in H\Rightarrow H',\ s\not\models a$ by construction $\Rightarrow H',\ s\models \sim a$ by model condition 2.

Induction: (We shorten the notation by dropping the H and H' and using $\in$ and $\models$ to distinguish between the notions.)

$$p\vee q\in s\Rightarrow p\in s \text{ or } q\in s\Rightarrow s\models p \text{ or } s\models q \text{ by induction}$$
$$\Rightarrow s\models p\vee q \text{ by 3.}$$

$\sim(p\vee q)\in s$. i.e. $\sim p\wedge\sim q\in s\Rightarrow \sim p\in s$ and $\sim q\in s\Rightarrow s\models\sim p$ and $s\models\sim q$

by induction $\Rightarrow s\not\models p$ and $s\not\models q\Rightarrow s\not\models(p\vee q)\Rightarrow s\models\sim(p\vee q)$ by 2. and 3.

$$\forall X\, p\in s\Rightarrow \text{for all successors } t_i,\ p\in t_i\Rightarrow t_i\models p \text{ by induction}$$
$$\Rightarrow s\models\forall X\, p$$

and similarly for the other modalities. ∎

A semantic tableau is a systematic search for a Hintikka structure. The tableau will be constructed as a tree of *nodes*. Each node is labelled with a set of formulas derived from the original formula p_0. Later we will show how to obtain a Hintikka structure from a successful tableau construction.

Notation. If n is a node of a tableau then U_n is the set of formulas labelling n. It is usually convenient to write $p\in n$ for $p\in U_n$. A further convenience is to replace the original formula p_0 by $p_0\wedge\forall G\exists X(\text{true})$ to ensure that the resulting structure is non-ending.

Tableau Construction. Let p_0 be a $\mathcal{UB}$ formula. Let n_0 be the root of T and let $U_{n_0}=\{p_0\}$. T, the tableau for p_0, is constructed inductively by applying the following rules to nodes n which are leaves of T.

R_α: If $\alpha\in U_n$ then create a son n_1 of n and define:

$$U_{n_1}=U_n\cup\{\alpha_1,\alpha_2\}.$$

R_β: If $\beta \in U_n$ then create two sons n_1 and n_2 of n and define:

$$U_{n_i} = U_n \cup \{\beta_i\}, \quad i = 1, 2.$$

The following rule is applied only when any application of the α/β rules would not generate a new node.

R_X: Let $V_n = \{\exists X p_1, \ldots, \exists X p_k, \forall X q_1, \ldots, \forall X q_m\}$ be the set of nexttime formulas in n. *Create* k sons n_i, $i = 1, \ldots, k$ of n and define:

$$U_{n_i} = \{p_i, q_1, \ldots, q_m\}.$$

Since we assumed that p_0 contains a conjunct $\forall G \exists X(\text{true})$, it follows that $k, m > 0$.

Notation. If the $R_\alpha/R_\beta/R_X$ rule was applied at node n then n is called an $\alpha/\beta/X$-node.

The construction of T is made finite by observing the following two termination rules.

1. If $p \in n$ and $\sim p \in n$ then this node is called *closed* and is not expanded further.

2. If a node m is about to be created as a son of an X-node n and there is an ancester n' of n such that n' is an immediate descendent of an X-node and $U_{n'} = U_m$ then do not create m but instead connect n to n' with a "feedback" edge.

The following lemma shows that the construction of T must terminate.

Lemma 4.2. *Let T be a tableau for p_0. Then for some (small) positive c, the number of distinct formulas that can appear in nodes of T is at most $c|p_0|$, where $|p_0|$ is the length of p_0.*

Proof. All formulas in nodes of T are subformulas of p_0 or negations or duals of subformulas or are subformulas prefixed by $\forall X$ or $\exists X$.

We now describe a marking algorithm which is designed to check if the tableau resulting from the construction can be made into a Hintikka structure. The algorithm is described in two parts. First, we describe a simple algorithm that will turn out to be incomplete and then we show the (substantial) changes that must be added in order to complete it.

Marking Algorithm

M1. Mark every closed leaf (node containing p and $\sim p$).
M2. If n is an α-node and n_1 is marked then mark n.
M3. If n is a β-node and both n_1 and n_2 are marked then mark n.
M4. If n is an X-node and some n_i is marked then mark n.

Define a structure as follows. Let S, the set of states, be the unmarked X-*nodes of* T. Call n a *pre-state* if n is the root or was created by the application

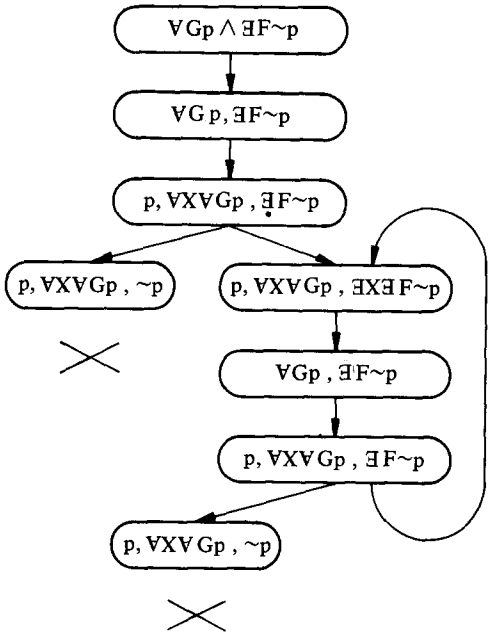


Fig. 1

of the X-rule to a state (X-node). Let sRt if there is a path $(s, n_0, \ldots, n_{k-1}=t)$ in T where $n_0, \ldots, n_{k-1}$ are α-nodes or β-nodes, $a \in P(s)$ iff $a \in s$.

It is easy to see that the resulting structure satisfies H1, H2, H3 and H4a-b but not necessarily H4c-d. For example, consider the tableau for $\forall Gp \wedge \exists F \sim p$ shown in Fig. 1. Every finite branch closes because any attempt to assign $\sim p$ to a state is inconsistent with $\forall Gp$ but there is an open branch which corresponds to always putting off to tomorrow the task of fulfilling $\exists F \sim p$.

Let T be a finite tableau whose unmarked nodes N satisfy: if $n \in N$ is an $\alpha/\beta/X$-node then the only son/ at least one son/ all sons of n are in N. Let us call a formula a future formula if it is of the form $\forall Fr$, $\forall X \forall Fr$, $\exists Fr$, or $\exists X \exists Fr$. We define a ranking algorithm which will assign a positive rank $\rho(q, n)$ to some occurrences q of future formulas in n.

Ranking Algorithm. Let $q \in n$ be a future formula which has not yet been ranked.

R 1. If n is a β-node *for* $q \in \{\forall Fr, \exists Fr\}$ and $r \in n_1 \in N$ then set $\rho(q, n)=1$.

R 2. If n is a β-node *for* $q \in \{\forall Fr, \exists Fr\}$ and $\rho(Xq, n_2)$ is defined then set $\rho(q, n)=\rho(Xq, n_2)+1$, where $Xq=\forall X \forall Fr/\exists X \exists Fr$ if $q=\forall Fr/\exists Fr$, resp.

R 3. If n is a β-node *not for* q and $\rho(q, n_i)$ is defined for some $i=1, 2$, then set $\rho(q, n)=\rho(q, n_i)+1$.

R 4. If n is an α-node and $\rho(q, n_1)$ is defined, then set $\rho(q, n)=\rho(q, n_1)+1$.

R 5 $\forall$. If n is an X-node, $q=\forall X \forall Fr$ and $\rho(\forall Fr, n_i)$ is defined for all n_i, then set $\rho(q, n)=\max_i(\rho(\forall Fr, n_i))+1$.

R5∃. If n is an X-node, $q=\exists X \exists F r$ and for some son n_i, $\exists F r \in n_i$ and $\rho(\exists F r, n_i)$ is defined, then set $\rho(q,n)=\rho(\exists F r, n_i)+1$.

Since the number of future formulas in N is finite and each is ranked at most once, the ranking algorithm must terminate.

Lemma 4.3. *(i)* $\rho(q,n)=1$ *iff* n *is a* β*-node for* q *and* $r \in n_1$.

Otherwise, if $\rho(q,n)>1$:

(ii) if n *is a* β*-node for* q *then* $\rho(X q, n_2)<\rho(q,b)$;

(iii) if n *is a* β*-node not for* q *(or an* α*-node) then* $\rho(q,n_i)<\rho(q,n)$ *for some (the only) son* n_i *of* n;

(iv) if n *is an* X*-node then* $\rho(\forall F r, n_i)<\rho(\forall X \forall F r, n)$ *for every son* n_i *of* n *and* $\rho(\exists F r, n_i)<\rho(\exists X \exists F r, n)$ *for the son* n_i *of* n *corresponding to* $\exists X \exists F r$.

Proof. Immediate from the definition of ρ. ∎

We now revise the marking algorithm as follows.

Marking Algorithm. Apply M1–M4 as long as possible.

At this stage the unmarked nodes form a structure N as described above.

M5. Apply the ranking algorithm to N. Mark each node containing an unranked occurrence of a future formula.

Of course, M5 can cause the structure to change so that M1–M4 again become applicable. However, since each stage marks at least one node, the algorithm must terminate, in fact, in time polynomial in the size of the tableau.

Theorem 4.4. n_0 *the root of* T, *is not marked iff there is a (finite) Hintikka structure for* $p_0 \in n_0$. *Hence, by Theorem 4.1 there is a decision procedure for* $\mathcal{UB}$ *and* $\mathcal{UB}$ *has the finite model property.*

Proof. $\Leftarrow$: This will follow from the completeness theorem to be given in the next section.

$\Rightarrow$: From the unmarked nodes of T we can form a structure as described before. However, this is still not a Hintikka structure. The problem can be demonstrated by the formula $r \equiv \forall G(\forall F(p \wedge q) \wedge \forall F(p \wedge \sim q) \wedge \forall F(\sim p \wedge q))$. The structure resulting from the tableau construction is shown in Fig. 2. The branch $s_0, s_1, s_0, s_1, \ldots$ is not fulfilling for $\forall F(\sim p \wedge q)$. The reason is that branching on β-nodes sometimes gives too much choice. This can be corrected by "unwinding" the structure as shown in Fig. 2 to force every infinite path to take both sons of a β-node.

We construct a tableau T' whose nodes are instances of nodes of T. Denote instances of $n \in T$ by $n', n'' \in T'$, etc. In the remainder of the proof, "nodes" refers only to unmarked nodes.

n'_0 is the root of T'. If n' is a leaf of T' then extend T' as follows.

Notation. $\pi(n'_i, n'_j)$ is a path from n'_i down to n'_j (not including n'_j) in T'. An *alternative node* is any β-node with two (unmarked) sons.

W1. If n is not an alternative node then for every son n_i of n let n'_i be a son of n' in T'.

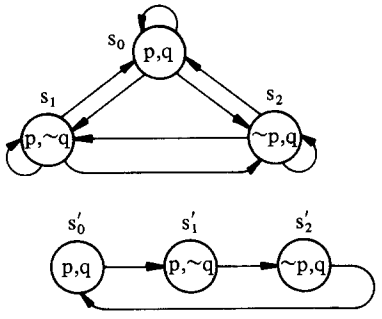


Fig. 2

W2. If n is an alternative node, let k be the number of instances of n in $\pi(n'_0, n')$.

(i) If $k=0$ then let n'_1 (arbitrarily) be the son of n' in T'.

(ii) If $k>0$ then if $n'_1(n''_2)$ was the son of n'' taken at the $k-1$'st instance n'' of n then let n'_2 (n'_1) be the son of n' in T'.

That is, we alternate our choices.

W3. However, if n has a previous instance n'' in $\pi(n'_0, n')$ and every alternative node which has an instance in $\pi(n'', n')$ has at least *two* instances in $\pi(n'', n')$ then identify n' with n''.

It is easy to see that the "unwinding" must terminate since there are a finite number of alternative nodes.

The following lemma is immediate from the construction.

Lemma 4.5. *Let b' be an (infinite) branch in T'. Let $n \in T$ be an alternative node which has an infinite number of instances $n', n'', \ldots$ in b'. Then each son n_i of n in T has an infinite number of instances in b'.*

Furthermore, it is trivial that the structure that can be defined from T' is a Hintikka structure except possibly for H4c-d. Thus Theorem 4.4 follows from the following lemmas.

Lemma 4.6. *Let $\forall F r \in n' \in T'$. Then all n'-branches are fulfilling.*

Lemma 4.7. *Let $\exists F r \in n' \in T'$. Then some n'-branch is fulfilling.*

Proof of Lemma 4.6. Suppose that there was an (infinite) nonfulfilling n'-branch $b' \in T'$. By the construction of T and T', $q = \forall F r$ or $\forall X \forall F r \in m'$ for all $m' \in b'$. Since each m' is an instance of an (unmarked) node m in T, $\rho(q, m)$ is defined.

Let $b' = (m'_1, m'_2, \ldots)$ and define $\rho_i = \rho(q, m_i)$ where m'_i is an instance in T' of m_i in T. Define $\rho^0 = \min\{\rho \mid \rho \text{ appears infinitely often in } (\rho_1, \rho_2, \ldots)\}$. Since the ranking algorithm terminates, there are a finite number of ranks assigned and thus some rank must appear i.o. and there must be a minimum such rank.

Since T is finite, there are a finite number of nodes $m \in T$ (perhaps only one) such that $\rho(q, m) = \rho^0$. Since ρ^0 appears i.o. in b, there must be a node m such that $\rho(q, m) = \rho^0$ and instances of m appear i.o. in b'. We now derive a contradiction to the minimality of ρ^0 at m.

$\rho^0 \neq 1$ since that would imply by Lemma 4.3 that m is a β-node for $\forall Fr$ and m_1 which contains r is not marked. But since m appears i.o. in b' by Lemma 4.5, so does m_1 contradicting the assumption that b is nonfulfilling.

Thus $\rho^0 > 1$. By Lemma 4.3:

If m is an α-node then an instance of its only son m_1 is in b' and satisfies $\rho(q, m_1) < \rho(q, m) = \rho^0$.

If m is a β-node then by Lemma 4.5 an instance of each (unmarked) son m_1 appears i.o. in b', in particular the m_i such that $\rho(q, m_i) < \rho(q, m) = \rho^0$.

If m is an X-node then $\rho(q, m_i) < \rho(q, m) = \rho^0$ for an instance of every son m_i of m, in particular the one following m' in b'.

Proof of Lemma 4.7. Let us define an n'-branch b' in T'. Note that in T' only the X-nodes offer a choice and thus we specify b' by requiring that if $m' \in b'$ is an X-node and $\exists X \exists Fr \in m'$ then choose the son of m' that contains $\exists Fr$. Since $\exists Fr \in n'$, either b' is fulfilling, or $q = \exists Fr$ or $\exists X \exists Fr \in m'$ for all $m' \in b'$.

If b' is not fulfilling then we proceed to define ρ^0 as in the previous lemma and derive a contradiction. The only difference is that if m is an X-node then we have (fortuitously) chosen the son m_i such that $\rho(\exists Fr, m_i) < \rho(\exists X \exists Fr, m) = \rho^0$ by Lemma 4.3.

5. Completeness

We now show that if a node n is marked by the marking algorithm then the conjunction of the formulas in U_n must be unsatisfiable because its negation is provable. If $U_n = \{p_i\}$, we define the *associated formula* of n, $\mathbf{af}_n$ to be $\sim \bigwedge p_i$. In particular, if n_0 is marked, then p_0 is unsatisfiable so $\vdash \sim p_0$. (It is easy to show that the proof system is sound).

In a typical proof of completeness by the tableau method ([15]), one shows that $\vdash \mathbf{af}_n$ for every leaf and that provability is preserved as one ascends the tree to the root. For nodes marked by M1–M4, the nexttime operator greatly simplifies the proof. For M5 we need a more difficult construction to show how the induction axioms must be used. These proofs were inspired by Pratt's work [13]. In practice, the tableau method for completeness is easy to use and the proofs of the theorems in Sect. 3 can be discovered by mechanical application of the tableau method.

Lemma 5.1. *If n is a leaf then* $\vdash \mathbf{af}_n$.

Proof. $p \in n$ and $\sim p \in n$. But $\vdash \sim(p \wedge \sim p)$ by R1 so $\vdash \mathbf{af}_n$ by dilution. (By dilution is meant: if there are other formulas $q, r, \ldots$ in n then from $\vdash \sim(p \wedge \sim p)$ we get $\vdash \sim p \vee \sim \sim p$, $\vdash \sim p \vee \sim \sim p \vee \sim q \vee \sim r \vee \ldots$ and then $\vdash \sim(p \wedge \sim p \wedge q \wedge r \ldots)$.)

Lemma 5.2. *If n is an $\alpha/\beta/X$-node and* $\vdash \mathbf{af}_{n_1} / \vdash \mathbf{af}_{n_1}$ *and* $\vdash \mathbf{af}_{n_2} / \vdash \mathbf{af}_{n_i}$ *for some i, then* $\vdash \mathbf{af}_n$.

Proof. For α and β nodes the Lemma follows by simple propositional reasoning and T13–T14. For an X-node:

$\vdash \sim(p_i \wedge q_1 \wedge \ldots \wedge q_l)$	by assumption
$\vdash (q_1 \wedge \ldots \wedge q_l) \supset \sim p_i$	
$\vdash \forall X(q_1 \wedge \ldots \wedge q_l) \supset \forall X \sim p_i$	A2
$\vdash (\forall X\, q_1 \wedge \ldots \wedge \forall X\, q_l) \supset \forall X \sim p_i$	T9
$\vdash (\forall X\, q_1 \wedge \ldots \wedge \forall X\, q_l) \supset \sim \exists X\, p_i$	Def. of $\exists X$
$\vdash (\forall X\, q_1 \wedge \ldots \wedge \forall X\, q_l) \supset \sim \exists X\, p_1 \vee \ldots \vee \sim \exists X\, p_k$	dilution
$\vdash \sim(\forall X\, q_1 \wedge \ldots \wedge \forall X\, q \wedge \exists X\, p_1 \wedge \ldots \wedge \exists X\, p_k)$	
$\vdash \mathbf{af}_n$	dilution.

Let t be a node (which is a state) in T which was marked because $\exists X \exists Fr$ was not ranked. Let $[t]^*$ be the set of *states* accessible from t by choosing at an X-node the son corresponding to $\exists X \exists Fr$. Then for every $u \in [t]^*$, there is an unranked occurrence of $\exists X \exists Fr$ (by construction of T and the definition of ρ).

Let $Y_u = \{q \mid \forall X\, q$ is (a universal nexttime formula) in $U_u\}$.

Let $W_u = \bigwedge_{q \in Y_u} q$ and $W^t = \bigvee_{u \in [t]^*} W_u$. W^t is called the *invariant* of t.

Lemma 5.3 $\vdash W^t \supset \forall X\, W^t$.

Proof. Let n_u be the pre-state created in T for the $\exists X \exists Fr$ son of u. Then $U_{n_u} = Y_u \cup \{\exists Fr\}$ and any state in $[t]^*$ accessible from u is accessible only by paths through n_u. Let $[u]$ be the states (all in $[t]^*$) accessible from n_u by α- and β-nodes. Now for any node n, $\vdash \bigwedge U_n \supset (\bigwedge U_{n_1}) \vee (\bigwedge U_{n_2})$ as shown by the various theorems we have proved (e.g. T13: $\vdash \forall G p \supset (p \wedge \forall X \forall G p)$; T14: $\vdash \forall F p \supset (p \vee \forall X \forall F p)$). By what was noted above, $\exists X \exists Fr \in v$ for all $v \in [t]^*$ hence in $[u]$.

Thus $\vdash W_u \supset \bigvee_{v \in [u]} (\bigwedge (U_v - \{\exists X \exists Fr\}))$.

What we have shown is that the conjunction of the formulas in a pre-state (not including $\exists Fr$) implies at least one of the conjunctions of the formulas in the successor states (except that formula produced by $\exists Fr$). In particular it implies the conjunction of all the universal nexttime formulas in the state. Thus:

$\vdash W_u \supset \bigvee_{v \in [u]} (\bigwedge_{p \in Y_v} \forall X p)$	
$\vdash W_u \supset \bigvee_{v \in [u]} (\forall X (\bigwedge_{p \in Y_v} p))$	T9
$\vdash W_u \supset \bigvee_{v \in [u]} (\forall X\, W_v)$	Def. of W_v
$\vdash W^t \supset \bigvee_{u \in [t]^*} (\forall X\, W_u)$	by taking the disjunction over all $u \in [t]^*$
$\vdash W^t \supset \forall X (\bigvee_{u \in [t]^*} W_u)$	T10
$\vdash W^t \supset \forall X\, W^t$	Def. of Wt.

Lemma 5.4. *For all* $u \in [t]^*$, $\vdash W_u \supset \sim r$.

Proof. Without loss of generality we can assume that the β-rule was applied to $\exists Fr$ at the pre-state n_u to construct sons n_1 and n_2. Then $U_{n_1} = \{U_{n_u} - \{\exists Fr\}\} \cup \{r\} = Y_u \cup \{r\}$. But n_1 must be marked for otherwise $\rho(\exists Fr, n_u) = 1$ and then $\rho(\exists X \exists Fr, u) = 2$ contrary to the assumption that $\exists X \exists Fr$ was not ranked in u. By the induction hypothesis $\vdash \mathbf{af}_{n_1}$ which is $\vdash \sim(W_u \wedge r)$ or $\vdash W_u \supset \sim r$. ∎

Theorem 5.5. $\vdash \mathbf{af}_t$

Proof

1. $\vdash W^t \supset \sim r$ — Disjunction over all $u \in [t]^*$ on Lemma 5.4
2. $\vdash \forall G W^t \supset \forall G \sim r$
3. $\vdash W^t \supset \forall G W^t$ — A4 on Lemma 5.3
4. $\vdash W^t \supset \forall G \sim r$ — 2, 3
5. $\vdash W_t \supset W^t$ — $t \in [t]^*$
6. $\vdash W_t \supset \forall G \sim r$ — 4, 5

But this is $\vdash \mathbf{af}_{n_t}$ for the pre-state created from state t. By Lemma 5.2:

7. $\vdash \mathbf{af}_t$.

We now proceed to the case where a state t was marked because $\forall X \forall Fr \in t$ was not ranked.

Lemma 5.6. *If* $\forall X \forall Fr \in t$ *was not ranked then at that stage of the marking algorithm, there exists a set of unmarked nodes* S *such that:*

a) $t \in S$;
b) for all $m \in S$, $q = \forall Fr$ *or* $\forall X \forall Fr \in m$ *and* $\rho(q, m)$ *is undefined;*
c) if $m \in S$ *is an* α*-node then* $m_1 \in S$;
d) if $m \in S$ *is a* β*-node then all unmarked sons* m_i *of* m *are in* S.
e) if $m \in S$ *is an* X*-node then some son* m_i *of* m *is in* S.

Proof. Let $S_0 = t$. Obviously, a) and b) hold. By induction assume that S_i has been constructed and a) and b) hold for S_i. Let m be a node in S_i for which one of c)-e) does not hold. Form S_{i+1} by adding to S_i as many sons of m as possible consistent with a) and b). c)-e) now hold for $m \in S_{i+1}$. Let us check d)-e).

If m is a β-node for q then it must be that m_1 is marked. Otherwise we would have $\rho(q, m) = 1$ by R1 contrary to b). If $\rho(q, m_2)$ is defined or if m is a ρ-node *not* for q and either $\rho(q, m_1)$ or $\rho(q, m_2)$ is defined then so would $\rho(q, m)$ by R2-R3. To check e), observe that if m is an X-node and $\forall X \forall Fr$ is unranked then there must be an m_i son of m in which $\forall Fr$ is unranked. m_i is added to S.

Since the ranking algorithm is applied to a finite structure, this procedure must terminate after, say k stages. $S = S_k$ has the required properties. ∎

Let $\forall X \forall Fr \in t$ and S be as in Lemma 5.6. For a state $u \in S$, some pre-state $n_u \in S$ was selected by 5.6.e. Let this be the node constructed in T for $\exists X p_u$. Let $Y'_u = Y_u \cup \{p_u\}$ where Y_u is defined as before.

Let $Z_u = \bigwedge_{q \in Y'_u} q$ and $Z^t = \bigvee_{u \in S} Z_u$.

Lemma 5.7. $\vdash Z^t \supset \exists X Z^t$.

Proof. Like Lemma 5.3 except that T11 is used instead of T9 to deduce that

$$\vdash Z_u \supset \bigvee_{v \in [u]} \left(\bigwedge_{p \in Y_u} \forall X p \wedge \exists X p_u \right)$$

implies

$$\vdash Z_u \supset \bigvee_{v \in [u]} (\exists X (\bigwedge_{p \in Y_u} p \wedge p_u)). \quad \blacksquare$$

Lemma 5.8. *For all* $u \in S$, $\vdash Z_u \supset \sim r$.

Proof. By Lemma 5.6, b and d the son of the β-node for $\forall F r$ containing r must be marked. By induction we can conclude that

$$\vdash \sim \{ \bigwedge_{q \in Y'_u - \{\forall F r\}} q \wedge r \}$$
$$\vdash \sim \{ \bigwedge_{q \in Y'_u} q \wedge r \} \qquad \text{dilution}$$
$$\vdash Z_u \supset \sim r. \quad \blacksquare$$

Lemma 5.9. $\vdash \mathbf{af}_t$.

Proof

1.	$\vdash Z^t \supset \sim r$	Disjunction on Lemma 5.8
2.	$\vdash \exists G Z^t \supset \exists G \sim r$	E1
3.	$\vdash Z^t \supset \exists G Z^t$	E4 on Lemma 5.7
4.	$\vdash Z^t \supset \exists G \sim r$	2, 3
5.	$\vdash Z_t \supset \exists G \sim r$	$t \in S$ and 4
6.	$\vdash Z_t \supset \sim \exists G \sim r$	Since $\forall F r$ is a conjunct of Z_t
7.	$\vdash Z_t \supset \mathit{false}$	6, 7
8.	$\vdash \sim Z_t$	which is $\vdash \mathbf{af}_{n_t}$
9.	$\vdash \mathbf{af}_t$	Lemma 5.2

Thus we have shown

Theorem 5.10. *A1–A4, E1–E4, R1–R3 form a complete axiom system for* $\mathcal{UB}$.

6. The $\mathcal{UB}$ Semantics of Nondeterministic Programs

The utility of $\mathcal{UB}$ for proving the program properties so elegantly expressible in the language depends on the ability to restrict the class of possible models to the class of execution trees of a given program P. This is done by specifying a set of axioms which impose the structure of computation according to a given program on our general models. It may also be considered as specifying the temporal semantics of the programming language by connecting its syntactical constructs to transformations and developments in time.

In order to do this we extend our language by allowing predicates on variables.

We have three types of variables:

a) Computation variables $y_1, y_2, \ldots$ which are modified by the execution and vary from state to state.

b) Free variables $x_1, x_2, \ldots$ which remain constant in time and are used to express relations between values of computation variables in different instances. Thus

$$(y=x) \supset \exists F(y=f(x))$$

is the expression of the statement that there exists some computation and some state in it such that the value of y in this state is equal to f of the initial y.

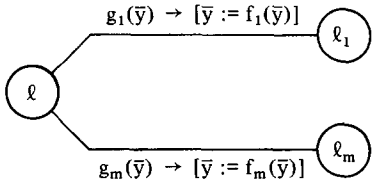
c) A variable π whose value at any state points to the segment of program yet to be executed.

Consider programs which are represented by transition graphs, $G=(N, E)$. The set N of nodes is called the set of locations. E is the set of edges each of which is labeled by a guarded instruction of the form

$$g(\bar{y}) \rightarrow [\bar{y} := f(\bar{y})]$$

with the meaning that this edge is enabled if the condition $g(\bar{y})$ is true and passing through the edge involves the assignment of $f(\bar{y})$ to $\bar{y}$.

We form our temporal semantics of such programs by letting π range over N (the location set) and forming for each node a semantic formula. Let a node $l \in N$ admit the following transitions:



Then we form the formula scheme

$$A_l\colon (\pi=l) \supset \left[\bigvee_{i=1}^{m} [g_i(\bar{y}) \wedge Q(l_i, f_i(\bar{y}))] \equiv \exists X\, Q \right]$$

Here $Q=Q(\pi, \bar{y})$ is an arbitrary predicate depending in general on the location variable π and the computation variables $\bar{y}$. It may also refer to free variables.

Note the presence of the '$\equiv$' connective which implies that this formula contains two implications. The first is stating that for every $i=1, \ldots, m$ such that g_i is true there is a successor state s in which $\pi_s=l_i$ and $\bar{y}_s=f_i(\bar{y})$. The other implication is a complementary statement saying that the only possible successor states are derived in this fashion. By using A_l as the axiomatic representation of a given nondeterministic program, it is possible to prove partial and total correctness of different types for nondeterministic programs. The system $\mathcal{UB}$ discussed here in great detail is only the propositional part of the full $\mathcal{UB}$ logic required in order to reason about concrete programs.

7. Discussion and Conclusions

In this paper we presented a unified branching time system which seems to enjoy the joint advantages of both linear time and branching time systems, in

being able to express and reason about the two basic types of termination, universal and existential. We have established the logical properties of the $\mathcal{UB}$ properties by presenting a decision algorithm and a complete deductive axiomatic system for the propositional fragment of the language. The decision procedure presented is obviously exponential.

This language must of course be compared with process logic languages such as PL [7] and its predecessors. The languages certainly can express any of the properties expressible in $\mathcal{UB}$ and many more. However, there is a price to pay for this expressibility which is the complexity of the languages. A sign of this is the fact that PL is nonelementary (has a nonelementary decision procedure) while $\mathcal{UB}$ is exponential.

Admittedly we do have six modal operators which is a disadvantage compared to simpler systems such as DX for linear time [5] or the corresponding branching time systems. On the other hand the formation rules of these operators are simple and uniform, and they do enable us to express most of the interesting program properties discussed in the literature.

Another advantage lost in the transition from linear to branching time is expressive completeness in the sense of [5]. Here the problem is inherent and cannot be remedied by the addition of one or two extra operators. This shown by the following:

Proposition 2. *No branching time temporal language with a finite number of modal operators can be expressively complete.*

This theorem, due to Gabbay (unpublished manuscript) is based on the following observations:

a) A temporal language with a finite number of operators can always be translated into a first order formula with a number of distinct variable names which is fixed for the language.

b) In a first order language it is easy to come up with formulas which need an arbitrarily large number of distinct variable names. Consider for example the statement:

There exist k time instants $t_1, \ldots, t_k$ no two of which are related.

This statement needs k variables for its expression for an arbitrary k. These formulas for sufficiently large k cannot therefore be expressed in any temporal logic.

While the proposed unified system is adequate for proving properties of nondeterministic programs it still falls short of linear temporal logic in being inadequate for proving properties of concurrent programs under the assumption of fairness. If we still adopt as basic model the tree of all possible computations, it necessarily contains some unfair computations as well. Consequently, we should be able to say that for every path in the tree, either the path represents an unfair computation or it satisfies some desirable property such as convergence. Such statements are inexpressible in $\mathcal{UB}$.

Since the writing of the first version of this paper there has been intensive activity in extending and refining the logic system and techniques presented here.

Most recent is the work reported in [3] which introduces CTL^+ - an extension of the $\mathcal{UB}$ system. While retaining the exponential complexity of $\mathcal{UB}$ it is possible to express in CTL^+ additional statements inexpressible in $\mathcal{UB}$.[5]

One of the novel features of the $\mathcal{UB}$ system is the introduction of the nexttime operators. In general, the nexttime operator is a relatively newcomer to temporal logic. Prior (in [14]; pp. 66-67) attributes the introduction of the next-instant operator to linear temporal logic to Scott. Kröger [8] was the first to use a temporal logic that included the nexttime operator for describing properties of nonterminating programs.

Acknowledgements. We would like to thank E.M. Clarke, Jr., A.E. Emerson and J.Y. Halpern for pointing out an error in the preliminary version. We would also like to thank L. Lamport and an anonymous referee for their comments and suggestions.

References

1. Abrahamson, K.R.: Modal logic of concurrent nondeterministic programs. Symposium on Semantics of Concurrent Computations, Lecture Notes in Computer Science 70, pp. 21-33. Berlin, Heidelberg, New York: Springer 1979
2. Ben-Ari, M., Manna, Z., Pnueli, A.: The temporal logic of branching time. Eight ACM Symposium on Principles of Programming Languages, Williamsburg, VA, pp. 164-176, 1981
3. Emerson, A.E., Halpern, J.Y.: Decision procedures and expressiveness in the temporal logic of branching time. 14-th ACM Symposium on Theory of Computing, San Francisco, CA, pp. 169-180, 1982
4. Floyd, R.W.: Nondeterministic algorithms. J. ACM **14**(4), 636-644 (1967)
5. Gabbay, D., Pnueli, A., Shelah, S., Stavi, J.: The temporal analysis of fairness. Seventh ACM Symposium on Principles of Programming Languages, Las Vegas, NE, pp. 163-173, 1980
6. Harel, D.: First Order Dynamic Logic, Lecture Notes in Computer Science 68. Berlin, Heidelberg, New York: Springer 1979
7. Harel, D., Kozen, D., Parikh, R.: Process Logic: expressiveness, decidability, completeness. 21-st IEEE Symposium on Foundation of Computer Science, pp. 129-142, 1980
8. Kröger, A.: A uniform logical basis for the description, specification and verification of programs. IFIP Working Conference on Formal Description of Programming Concepts, St. Andrews, Canada, 1977
9. Lamport, L.: "Sometime" is sometimes "not never". Seventh ACM Symposium on Principles of Programming Languages, Las Vegas, NE, pp. 174-183, 1980
10. Manna, Z.: Second order mathematical theory of computation. Second ACM Symposium on Theory of Computing, pp. 158-168, 1970
11. Manna, Z., Pnueli, A.: The modal logic of programs. Automata, Languages and Programming, Lecture Notes in Computer Science 79, pp. 385-409. Berlin, Heidelberg, New York: Springer 1979
12. Pnueli, A.: The temporal semantics of concurrent programs. Theor. Comput. Sci. **13**, 45-60 (1981)
13. Pratt, V.R.: A near optimal method for reasoning about action. J. Comput. Syst. Sci. **20**, 231-254 (1980)
14. Prior, A.: Past, Present and Future. Oxford University Press 1967
15. Rescher, N., Urquhart, A.: Temporal Logic. Berlin, Heidelberg, New York, Wien: Springer 1971
16. Smullyan, R.M.: First Order Logic. Berlin, Heidelberg, New York: Springer 1968

Received October 14, 1982/May 2, 1983

[5] A recent manuscript by Clarke and Emerson has shown how to use this logic to synthesize concurrent programs