DECISION PROCEDURES AND EXPRESSIVENESS IN THE TEMPORAL LOGIC OF BRANCHING TIME

E. Allen Emerson & Joseph Y. Halpern[1]

Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712

TR-85-29 November 1985

Reprinted from *Journal of Computer and System Sciences*.

[1]IBM Research Laboratory, 5600 Cottle Road, San Jose, CA 95193.

Decision Procedures and Expressiveness in the Temporal Logic of Branching Time*

E. ALLEN EMERSON†

Computer Sciences Dept., University of Texas, Austin, Texas 78712

AND

JOSEPH Y. HALPERN‡

IBM Research Laboratory, 5600 Cottle Road, San Jose, California 95193

We consider the computation tree logic (CTL) proposed in (*Sci. Comput. Programming* **2** (1982), 241–260) which extends the unified branching time logic (UB) of ("Proc. Ann. ACM Sympos. Principles of Programming Languages, 1981," pp. 164–176) by adding an until operator. It is established that CTL has the small model property by showing that any satisfiable CTL formulae is satisfiable in a small finite model obtained from the small "pseudo-model" resulting from the Fischer–Ladner quotient construction. Then an exponential time algorithm is given for deciding satisfiability in CTL, and the axiomatization of UB given in ibid. is extended to a complete axiomatization for CTL. Finally, the relative expressive power of a family of temporal logics obtained by extending or restricting the syntax of UB and CTL is studied.

1. INTRODUCTION

Temporal logic is a formalism for reasoning about correctness properties of concurrent programs [15, 13]. In practice, it has been found useful to have an until operator $p\ U\ q$ which asserts that q is bound to happen, and until it does p will hold (cf. [10]). In this paper we consider the computation tree logic (CTL) proposed by Clarke and Emerson [5] which extends the unified branching time logic (UB) of Ben-Ari, Manna, and Pnueli [4] by adding such an until operator. We give an exponential time algorithm for deciding satisfiability in CTL and extend the axiomatization of UB given in [4] to one for CTL.

* This is an expanded version of a paper with the same title which was given at the 14th Annual ACM Symposium on Theory of Computing, San Francisco, California, May 5–7, 1982.

† Partially supported by NSF Grant MCS79-08365.

‡ Partially supported by NSF Grant MCS80-10707 and a grant from the National Science and Engineering Research Council of Canada. Most of this work was done while a visiting scientist jointly at MIT and Harvard.

Our first step is to establish that CTL has the *small model property*: if a formula is satisfiable, then it is satisfiable in a small finite model. The standard way of proving such results for modal logics is to "collapse" a (possibly infinite) model by identifying states according to an equivalence relation of small finite index, and then showing that the resulting finite quotient structure is still a model for the formula in question. This technique is used, for example, by Fischer and Ladner to show that PDL has the small model property (cf. [9]). We show that any method of trying to prove the small model property directly by using a quotient construction must fail when applied to UB or CTL. However, we can also show that the Fischer–Ladner quotient structure obtained from a CTL model may be viewed as a small "pseudo-model" which contains enough information to be unwound into a genuine (and still small) model.

Both our algorithm for deciding satisfiability and our completeness proof are based on trying to construct this pseudo-model. Our approach is similar to that used in [3] to show the corresponding results for DPDL, which suggests that the pseudo-model phenomenon may be a general one which is applicable to a variety of temporal logics. We then reprove these results by using the fixpoint characterizations (cf. [6]) of the temporal operators to construct a tableau which may itself be considered a small pseudo-model. Our first method can be viewed as a "top-down" approach, while the tableau method is "bottom-up." Although both decision procedures given have the same worst-case complexity of exponential time (which is provably the best we can do), the tableau method is likely to be better in practice. (Another tableau-based algorithm for satisfiability in UB was proposed in [4]. However, that algorithm claims that certain satisfiable formulae are unsatisfiable. Ben-Ari [2] states that a corrected version is forthcoming.)

We also study the expressive power of temporal logics obtained by extending or restricting UB. In UB a path quantifier, either A ("for all paths") or E ("for some path"), is always paired with a *single* state quantiier, either F ("for some state"), G ("for all states"), or X ("for the next state"). Thus, the UB syntax allows the assertions EFp (for some computation path, there is a state on the path where p holds) and EGp (for some computation path, for all states on the path, p holds). If we extend the syntax to allow assertions such as $E[Fp \wedge Gq]$ (for some computation path, there is a state on the path where p holds and for all states on that same path, q holds), where a path quantifier is paired with a Boolean combination of state quantifiers, we obtain the language we call UB$^+$. Similarly, CTL$^+$ is obtained by extending CTL, to allow a path quantifier to prefix a Boolean combination of the state quantifiers F, G, X, or U. Finally, UB$^-$ is obtained by restricting the UB syntax to allow only the pairs EX and EF (AG and AX can be obtained by negation) and corresponds to the *nexttime* logic of Manna and Pnueli [14]. We show that these languages can be arranged in the following hierarchy of expressive power: $UB < UB < UB^+ < CTL \equiv CTL^+$.

The rest of the paper is organized as follows: Section 2 gives the syntax and semantics of CTL$^+$ (and by suitable restriction, of all the other languages). Section 3 shows why quotient constructions must fail for UB and CTL and defines the technical machinery of Hintikka structures (cf. [4]) and pseudo-Hintikka structures necessary for our constructions. In Sections 4, 5, and 6 the first proofs of the small model theorem, the decision procedure, and completeness of the axiom system, respectively, are given. These results are reestablished in Section 7 using tableau techniques. In Section 8 we give our expressibility results. Finally, in Section 9 we make some concluding remarks.

2. Syntax and Semantics

2.1. *Syntax*

We define below a language which extends UB and CTL in order to provide a framework for the expressiveness results of Section 8. We start with a set of *primitive* (or *atomic*) formulae $\Phi_0 = \{P, Q, \ldots\}$. We then inductively define a set of state formulae and a set of path formulae:

(1) Each primitive formula is a state formula.

(2) If p, q are state formulae, then so are $(p \wedge q)$ and $\neg p$.

(3) If p is a state formula, then Fp and Xp are path formulae (which intuitively say that at some state (resp. the next state) on the path p holds).

(4) If p is a path formula then Ep is a state formula (which says some path satisfies p).

(5) If p is a path formula then Ap is a state formula (which says all paths satisfy p).

(6) If p, q are state formulae then $(p\, U\, q)$ is a path formula (which says there is some state on the path which satisfies q, and all states before it satisfy p, i.e., p holds until q).

(7) If p, q are path formulae, then so are $p \wedge q$ and $\neg p$.

We use the abbreviations $p \vee q$ for $\neg(\neg p \wedge \neg q)$, $p \Rightarrow q$ for $\neg p \vee q$, $p \equiv q$ for $(p \Rightarrow q) \wedge (q \Rightarrow p)$, and Gp for $\neg F \neg p$.

The *size* of formula p, written $|p|$, is its length over the alphabet $\{\neg, \wedge, (,), E, A, F, U, X\} \cup \Phi_0$. The state formulae generated by rules (1)–(4), rules (1)–(5), and rules (1)–(6) correspond exactly to UB^-, UB, and CTL, respectively. Define UB^+ to be the state formulae generated by rules (1)–(5), (7) and CTL^+ to be the state formulae generated by rules (1)–(7).

2.2. *Structures*

A *structure* $M = (S, L, R)$ consists of a set S of *states*, an *assignment* L of formulae to states, and a binary *relation* $R \subseteq S \times S$. (Think of $L(s)$ as the formulae true at state s.) A *path* is a sequence $(s_0, s_1, \ldots)$ of states such that $(s_i, s_{i+1}) \in R$ that is maximal (i.e., either infinite or whose last state has no R-successor). We can view a structure as a labelled directed graph whose nodes are the states. Node s is labelled by the formulae in $L(s)$, and there is an arc from s to t iff $(s, t) \in R$. The *size* of a structure $M = (S, L, R)$ is the cardinality of S.

2.3. *Models*

Given a structure $M = (S, L, R)$, we want to define the notion of *truth* in M via the relation $\models$. Given a state s (resp. path x) in M, and a state formula p (resp. path formula p') we write $M, s \models p$ (resp. $M, x \models p'$), which means p is true of state s (p' is true of path x) in M. We define $\models$ inductively as follows:

(M1) For a primitive formula P, $M, s \models P$ iff $P \in L(s)$.

(M2) If p, q are state formulae, $M, s \models p \wedge q$ iff $M, s \models p$ and $M, s \models q$; $M, s \models \neg p$ iff $M, s \not\models p$.

(M3) If $x = (s_0, s_1, ...)$ is a path, then $M, x \models Fp$ iff for some s_i on x, $M, s_i \models p$; $M, x \models Xp$ iff $M, s_1 \models p$.

(M4) $M, s \models Ep$ if for some path x starting at s, $M, x \models p$.

(M5) $M, x \models Ap$ if for all paths x starting at s, $M, x \models p$.

(M6) $M, x \models (p\ U\ q)$ if for some initial prefix $(s_0, ..., s_k)$ of x, $M, s_k \models q$ and $M, s_i \models p$ for all $i < k$.

(M7) If p, q are path formulae, $M, x \models p \wedge q$ iff $M, x \models p$ and $M, x \models q$; $M, x \models \neg p$ iff $M, x \not\models p$.

A *model* is a structure $M = (S, L, R)$ such that R is total and for all states $s \in S$ and all state formulae p, we have $M, s \models p$ iff $p \in L(s)$. Note that in a model $M = (S, L, R)$, L is completely determined by the primitive formulae in $L(s)$.

2.4. *Remark*

For technical reasons, we have used L here, an assignment of formulae to states, rather than the more usual π, an assignment of states to formulae (cf. [3, 4]). It is easy to see that this slight change does not affect any of the results. Of course, $\models$ is still defined in the usual way. We also follow [4, 6] in requiring that in a model R be total. This restriction can be removed without affecting any of the main theorems (cf. Sect. 6.3).

2.5. DEFINITION. A state formulae p is *satisfiable* (resp. *valid*) iff for some model (resp. all models) $M = (S, L, R)$ and some (resp. all) $s \in S$, $M, s \models p$. Similarly for path formulae. We write $\models p$ if p is valid. Note that p is satisfiable iff $\neg p$ is not valid.

The following lemma shows that the temporal operators may be viewed as fixpoints of appropriate functionals (see [6]). For example, EFp is a fixpoint of $f(z) = p \vee EXz$. This forms the basis of the tableau construction of Section 7.

2.6. LEMMA. *The following formulae are valid:*

(1) $\models Fp \equiv (true\ U\ p)$

(2) $\models EFp \equiv p \vee EX(EFp)$

(3) $\models AFp \equiv p \vee AXAFp$
(4) $\models E(p\, U\, q) \equiv q \vee (p \wedge EXE(p\, U\, q))$
(5) $\models A(p\, U\, q) \equiv q \vee (p \wedge AXA(p\, U\, q))$.

Proof. Immediate from the definitions in 2.3. Note that by part (1), it follows that (2) and (3) are just special cases of (4) and (5) obtained by taking p to be *true*. We include the special cases for UB here and in future lemmas and theorems to show that our techniques apply directly to UB. ∎

For the next five sections we focus our attention on UB and CTL.

3. Hintikka Structures and the Quotient Construction

In order to help us obtain a decision procedure and axiomatization for CTL, we use *Hintikka structures*, which are based on Smullyan's *semantic tableaux* (cf. [18]). Roughly speaking, a Hintikka structure is a structure where the formulas of $L(s)$ "true" at a state s satisfy certain consistency conditions which seem weaker than those required for a model, but, in a certain sense made precise in Proposition 3.2, are equivalent.

3.1. Definition. A *Hintikka structure* (for p_0) is a structure $M = (S, L, R)$ with R total (and $p_0 \in L(s)$ for some $s \in S$) which satisfies the following constraints:

(H1) $\neg p \in L(s) \Rightarrow p \notin L(s)$
(H2) $\neg\neg p \in L(s) \Rightarrow p \in L(s)$
(H3) $p \wedge q \in L(s) \Rightarrow p, q \in L(s)$
(H4) $\neg(p \wedge q) \in L(s) \Rightarrow \neg p \in L(s)$ or $\neg q \in L(s)$
(H5) $EFp \in L(s) \Rightarrow p \in L(s)$ or $EXEFp \in L(s)$
(H6) $\neg EFp \in L(s) \Rightarrow \neg p, \neg EXEFp \in L(s)$
(H7) $AFp \in L(s) \Rightarrow p \in L(s)$ or $AXAFp \in L(s)$
(H8) $\neg AFp \in L(s) \Rightarrow \neg p, \neg AXAFp \in L(s)$
(H9) $E(p\, U\, q) \in L(s) \Rightarrow q \in L(s)$ or $p, EXE(p\, U\, q) \in L(s)$
(H10) $\neg E(p\, U\, q) \in L(s) \Rightarrow \neg q, \neg p \in L(s)$ or $\neg q, \neg EXE(p\, U\, q) \in L(s)$
(H11) $A(p\, U\, q) \in L(s) \Rightarrow q \in L(s)$ or $p, AXA(p\, U\, q) \in L(s)$
(H12) $\neg A(p\, U\, q) \in L(s) \Rightarrow \neg q, \neg p \in L(s)$ or $\neg q, \neg AXA(p\, U\, q) \in L(s)$
(H13) $EXp \in L(s) \Rightarrow \exists t((s, t) \in R$ and $p \in L(t))$
(H14) $\neg EXp \in L(s) \Rightarrow \forall t((s, t) \in R \Rightarrow \neg p \in L(t))$
(H15) $AXp \in L(s) \Rightarrow \forall t((s, t) \in R \Rightarrow p \in L(t))$
(H16) $\neg AXp \in L(s) \Rightarrow \exists t((s, t) \in R$ and $\neg p \in L(t))$
(H17) $EFp \in L(s) \Rightarrow$ for some path x starting at s and some state t on x, $p \in L(t)$
(H18) $AFp \in L(s) \Rightarrow$ for all paths x starting at s and some state t on x, $p \in L(t)$
(H19) $E(p\, U\, q) \in L(s) \Rightarrow$ for some path x starting at s, some state t on x, and all states t' before t on x, $q \in L(t)$ and $p \in L(t')$

(H20) $A(p\ U\ q) \in L(s) \Rightarrow$ for all paths x starting at s, for some t on x and all states t' before t on x, $q \in L(t)$ and $p \in L(t')$.

3.2. PROPOSITION. (cf. [4, Theorem 1]). *A model for p is a Hintikka structure for p. Conversely, a Hintikka structure (S, L, R) can be extended to a model (S, L', R), where $L(s) \subseteq L'(s)$ for all $s \in S$.*

Proof. A model for p is clearly a Hintikka structure for p. Conversely, given a Hintikka structure $M = (S, L, R)$, define $L'(s) = \{p \mid M, s \models p\}$, where $\models$ is as defined in Section 2.3. $M' = (S, L', R)$ is clearly a model. We can now show by induction on the structure of q that if $q \in L(s)$, then $q \in L'(s)$, and so $L(s) \subseteq L'(s)$. We leave details to the reader. ∎

3.3. DEFINITION. We define the Fischer–Ladner closure (cf. [9]) of a CTL formula p_0, which for technical reasons we close under negation. Let $H(p_0)$ be the least set of formulae containing p_0 such that

(1) $\neg p \in H(p_0) \Rightarrow p \in H(p_0)$
(2) $p \wedge q \in H(p_0) \Rightarrow p, q \in H(p_0)$
(3) $EFp \in H(p_0) \Rightarrow p, EXEFp \in H(p_0)$
(4) $AFp \in H(p_0) \Rightarrow p, AXAFp \in H(p_0)$
(5) $E(p\ U\ q) \in H(p_0) \Rightarrow q, p, EXE(p\ U\ q) \in H(p_0)$
(6) $A(p\ U\ q) \in H(p_0) \Rightarrow q, p, AXA(p\ U\ q) \in H(p_0)$
(7) $EXp \in H(p_0) \Rightarrow p \in H(p_0)$
(8) $AXp \in H(p_0) \Rightarrow p \in H(p_0)$.

Let $FL(p_0) = H(p_0) \cup \neg H(p_0)$ (where $\neg H(p_0) = \{\neg p \mid p \in H(p_0)\}$).

3.4. LEMMA (cf. [9, 4, 3]). $|FL(p)| \leqslant 2|p|$.

Proof. An easy induction on $|p|$ shows that $|H(p)| \leqslant |p|$. The result follows immediately. ∎

3.5. *The Quotient Construction*

One elegant way to establish that a propositional temporal logic has the small model property is to use a quotient construction. Let $M = (S, L, R)$ be a model of p, let H be a set of formulae, and let $\equiv_H$ be an equivalence relation on S defined via $s_1 \equiv_H s_2$ iff for all $q \in H$, $M, s_1 \models q$ iff $M, s_2 \models q$. Use $[s]$ to denote $\{t \in S \mid t \equiv_H s\}$. Then the quotient structure of M by $\equiv_H$ is defined to be the structure $M/\equiv_H = (S', L', R')$, where $S' = \{[s] \mid s \in S\}$, $R' = \{([s], [t]) \mid (s, t) \in R\}$, and $L'([s]) = L(s) \cap H$. We remark that although $|FL(p)| \leqslant 2|p|$, there are at most $2^{|p|}$ $\equiv_{FL(p)}$ equivalence classes.

Fischer and Ladner showed that for a PDL formula p, if M is a model for p, then $M/\equiv_{FL(p)}$ is a Hintikka structure for p with the property that satisfiability is preserved for formulas in $FL(p)$; i.e., for $q \in FL(p)$, $M, s \models q$ iff $M/\equiv_{FL(p)}, [s] \models q$. Unfortunately no such quotient construction will directly show that UB (or CTL)

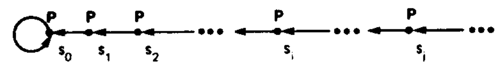

FIGURE 1

has the small model property. The following theorem is analogous to those in [16, 18] showing that the property of looping is not expressible in PDL.

3.6. THEOREM. *For a finite set H of* UB (*or* CTL) *formulas, the operation of forming the quotient structure by $\equiv_H$ does not preserve satisfiability for the formula AFP. In particular, there is a model M which satisfies AFP such that for every finite set H, $M/\equiv_H$ is* not *a Hintikka structure for AFP.*

Proof. Note that in the structure depicted in Fig. 1, where $M, s_0 \models P$ and $M, s_i \models \neg P$ for $i>0$, we clearly have $M, s_i \models AFP$ for all $i \geqslant 0$. Yet, if H is any finite set, we must have $s_i \equiv_H s_j$ for some $i>j>0$. Thus $M/\equiv_H$ looks like Fig. 2. Now if $AFP \notin H$, then clearly $M/\equiv_H$ cannot be a Hintikka structure for AFP. And if $AFP \in H$, then we must have $AFP \in L([s_i])$, which violates (H18), so again $M/\equiv_H$ is not a Hintikka structure for AFP. ∎

Nonetheless, the quotient structure $M/\equiv_{FL(p)}$ does provide useful information. It is easy to check that $M/\equiv_{FL(p)}$ satisfies all the constraints of Definition 3.1 except possibly (H18) and (H20) (which, as shown in the proof of Theorem 3.6, it does not, in general, satisfy). Instead, $M/\equiv_{FL(p)}$ satisfies another important property which will allow us to view it as a "pseudo-model" that can be "unwound" into a genuine model. To make these ideas precise, we need the following definitions.

3.7. DEFINITION. Given a directed acyclic graph (dag), an interior (resp. frontier) node of the graph is one which has (resp. does not have) a successor. The root of a dag is the unique node (if it exists) from which all other nodes are reachable. A *fragment* $M=(S, L, R)$ is a structure whose graph is a finite dag whose interior nodes satisfy (H1)–(H16), and whose frontier nodes satisfy (H1)–(H12). Given $M_1=(S_1, L_1, R_1)$ and $M_2=(S_2, L_2, R_2)$, we say M_1 is *contained in* M_2 and write $M_1 \subseteq M_2$ iff $S_1 \subseteq S_2$, $L_1 = L_2 | S_1$ and $R_1 \subseteq R_2$. We say M_1 is *embedded in* M_2, and write $M_1 \leqslant M_2$, iff $M_1 \subseteq M_2$ and $(s_1, s_2) \in R_2 \cap (S_1 \times (S_2 - S_1))$ implies s_1 has no R_1-successor (i.e., the only arcs from nodes of M_1 to nodes of M_2 begin at frontier nodes of M_1).

3.8. LEMMA. *Let $M=(S, L, R)$ be a model for p_0, and let $M' = M/\equiv_{FL(p_0)} = (S', L', R')$. Suppose AFq (resp. $A(p\, U\, q)) \in L'([s'])$. Then there is a fragment N*

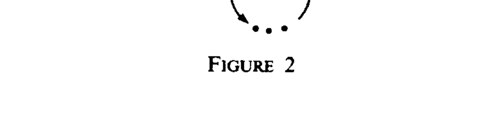

FIGURE 2

rooted at $[s']$ *contained in* M' *such that for all the frontier nodes* t *of* N, $q \in L'(t)$ (*resp. and for all interior nodes* u *of* N, $p \in L'(u)$).

Proof. We give the proof for AFq. The proof for $A(p\,U\,q)$ is similar. We first assume that in the original structure M, each node has a finite number of successors. (The case where some node has an infinite number of successors is considered at the end of the proof.)

Choose $s \in [s']$. Then it is easy to see that embedded in M there is a fragment rooted at s of the form claimed by the lemma. Simply take all nodes on paths starting at s up to (and including) the first node containing q in its label. This must be a finite dag; otherwise, by Koenig's lemma, (H18) would be violated.

If the labels on the nodes are all distinct, then this fragment is also contained in M' and we are finished. If not, we will systematically eliminate "duplicate" nodes from this fragment until we finally obtain a fragment which is contained in M'.

We proceed as follows (see Fig. 3). Define the *depth* of a node t, $d(t)$, in a dag as the length of the longest path from the root to t. Then suppose that we have two distinct nodes t_1 and t_2 with identical labels, $L'(t_1) = L'(t_2)$, such that $d(t_1) \geqslant d(t_2)$. We let the deeper node t_1 replace the shallower node t_2 to get a new fragment; i.e., we replace each arc (u, t_2) by the arc (u, t_1) and eliminate all nodes no longer reachable from the root, as shown in the diagram below. Note that t_2 itself is no longer reachable from the root, so it is eliminated.

The resulting graph is easily seen to still be a fragment rooted at s such that for all frontier nodes t, $q \in L'(t)$. We continue this process until the labels on all the nodes are distinct. This process must terminate after a finite number of steps since the original fragment was finite. The resulting fragment is contained in M' and meets the conditions of the lemma.

If the original structure M had one or more nodes with an infinite number of successors, we construct a structure M' with no such nodes as follows: For each node t and each formula of the form EXq' or $\neg AXq''$ in $L(t) \cap FL(p_0)$, choose an arc $(t, u) \in R$ such that $q' \in L(u)$ or $\neg q'' \in L(u)$, respectively. Eliminate the edges not chosen. Let the resulting relation be R'' and let $M'' = (S, L, R'')$. Each node of M'' has only a finite number of successors ($\leqslant |p_0|$), and it is easy to check that we can carry out the above construction using M'' instead of M since (H18) still holds

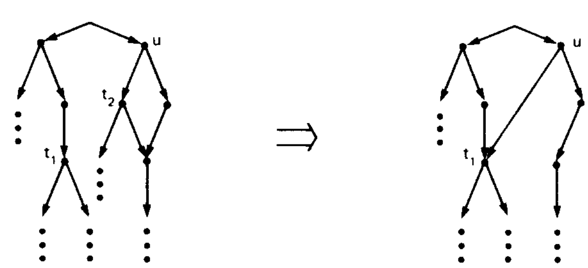

FIGURE 3

(although, in general, M'' is *not* a model for p_0 since the eliminated arcs may have been necessary for fulfillment of formulae such as EFp). ∎

3.9. DEFINITION. A pseudo-Hintikka structure (for p_0) is a structure $M = (S, L, R)$ with R total (such that $p_0 \in L(s)$ for some $s \in S$) which satisfies (H1)–(H17), (H19), and for all $s \in S$,

(H18′) $AFp \in L(s)$ implies there is a fragment N rooted at s contained in M such that for all frontier nodes t of N, $p \in L(t)$.

(H20′) $A(p\ U\ q) \in L(s)$ implies there is a fragment N rooted at s contained in M such that for all frontier nodes t of N, $q \in L(t)$, and for all interior nodes u of N, $p \in L(u)$.

4. A SMALL MODEL THEOREM FOR CTL

4.1. THEOREM. *Let p_0 be a* CTL *formula with $|p_0| = n$. Then the following are equivalent*:

(a) *p_0 is satisfiable.*

(b) *There is a pseudo-Hintikka structure for p_0 of size $\leqslant 2^n$.*

(c) *There is a Hintikka structure for p_0 of size $\leqslant n8^n$.*

(d) *p_0 is satisfiable in a model of size $\leqslant n8^n$.*

Proof. (a) ⇒ (b) follows from Lemma 3.8; (c) ⇒ (d) follows from the proof of Proposition 3.2; and (d) ⇒ (a) is immediate. It remains to prove (b) ⇒ (c). We need the following definition:

4.2. DEFINITION. Let M be a structure, s a state in M, and $p \in L(s)$, where p is an *eventuality formula*, i.e., p is of the form EFq, AFq, $E(q'\ U\ q)$, or $A(q'\ U\ q)$. We say p is *fulfilled* in M for s if (H17), (H18), (H19), or (H20), respectively, holds for p and s.

We will construct a finite Hintikka structure for p_0 by "unravelling" the pseudo-Hintikka structure M for p_0. For each node s of M and for each eventuality formula $p \in L(s)$, there is a fragment $DAG[s, p]$ which certifies that p is fulfilled for s. We show how to use these DAGs to construct for each node s of M, a fragment $FRAG[s]$ such that every eventuality formula in $L(s)$ is fulfilled within $FRAG[s]$. We then splice together these $FRAG$s to obtain the desired finite Hintikka structure. This is described in detail below.

The following lemma says that if an eventuality formula is not fulfilled in a fragment then the conditions required to fulfill it are propagated down to appropriate frontier nodes of the fragment. The proof is straightforward and is omitted here. (Note, however, that substantial use is made of the fact that a fragment is acyclic.)

4.3. LEMMA. *Let M be a fragment, s a state in M, and p an eventuality formula in $L(s)$. Either p is fulfilled in M for s or*

(a) *If p is of the form EFq (resp., $E(q'\ U\ q)$), then there is a path in M from s to a frontier node t such that $EFq \in L(t)$ (resp., $E(q'\ U\ q) \in L(t)$ and $q' \in L(t')$ for every state t' on the path). Moreover, if M is embedded in M' and EFq (resp. $E(q'\ U\ q)$) is fulfilled in M' for t, then EFq (resp. $E(q'\ U\ q)$) is fulfilled in M' for s.*

(b) *If p is of the form AFq (resp., $A(q'\ U\ q)$), then for every path in M from s to a frontier node t, either $q \in L(t')$ for some t' on the path, or $AFq \in L(t')$ for all t' on the path (resp., either $q \in L(t')$ for some t' on the path and $q' \in L(t'')$ for all t'' on the path from s before t', or $q', A(q'\ U\ q) \in L(t'')$ for every t'' on the path). Moreover, if M is embedded in M' and AFq (resp., $A(q'\ U\ q)$) is fulfilled in M' for all frontier nodes t of M, then AFq (resp. $A(q'\ U\ q)$) is fulfilled in M' for s.*

4.4. LEMMA. *Let M be a pseudo-Hintikka structure of size N_0, s a state of M, and p an eventuality formula in $L(s)$. Then we can find a fragment, $DAG[s, p]$, with $\leqslant N_0$ interior nodes and root s (i.e., labelled $L(s)$) in which p is fulfilled for s.*

Proof. That we can find $DAG[s, AFq]$ (resp. $DAG[s, A(q'\ U\ q)]$) follows directly from (H18′) (resp. (H20′)).

For $DAG[s, EFq]$, note that if $EFq \in L(s)$ then by (H17) we can find a path in M starting at s to some state t with $q \in L(t)$. Choose a shortest such path. Its length must be $\leqslant N_0$ (otherwise, some state must be repeated and there would be a shorter path). Take the path from s to t, and for every state other than t, add enough successors to ensure that (H13) and (H16) are satisfied. (Recall that interior nodes of a fragment must satisfy (H1)–(H16). The interior nodes on the path clearly satisfy all the properties besides (H13) and (H16). By adding on these successors, we ensure that they satisfy these properties too.) The resulting graph defines $DAG[s, EFq]$ and it is easy to check that all other conditions are satisfied. The construction for $DAG[s, E(q'\ U\ q)]$ is similar. ∎

In the proof of Lemma 4.5 we explain how to construct *FRAG*s from *DAG*s. The observations of Lemma 4.3 will allow us to "glue" together the fragments $DAG[s, p]$ constructed in Lemma 4.4 in such a way as to get a fragment where all the eventuality formulas of $L(s)$ are satisfied.

4.5. LEMMA. *Suppose s is a state in a pseudo-Hintikka structure M of size N_0. Then we can find a fragment, $FRAG[s]$, with $\leqslant |L(s)|\ N_0^2$ interior nodes and root s (i.e., labelled $L(s)$) such that all eventuality formulae in $L(s)$ are fulfilled for s in $FRAG[s]$.*

Proof. We construct $FRAG[s]$ in stages. Let $q_1, ..., q_k$ be a list of all eventuality formulae in $L(s)$. We will build a sequence of fragments $M_0 \leqslant M_1 \leqslant \cdots \leqslant M_k = FRAG[s]$ all with root s such that, for each j, M_j has at most jN_0^2 interior nodes, at most N_0 frontier nodes, and $q_1, ..., q_j$ are all fulfilled for s in M_j.

We define the sequence inductively. Let M_0 consist of s with just enough successors to ensure (H13) and (H16) are satisfied. M_{j+1} is obtained by extending the frontier of M_j as follows: If q_{j+1} is fulfilled for s in M_j, let $M_{j+1} = M_j$. Otherwise, suppose q_{j+1} is of the form EFq. Then, by Lemma 4.3, there is a frontier node t of M_j, with $EFq \in L(t)$. Let M'_{j+1} be the result of replacing t by (a copy of) $DAG[t, EFq]$. Since EFq is fulfilled for t in M'_{j+1}, by Lemma 4.3, EFq is fulfilled for s in M'_{j+1}. To ensure that M_{j+1} has at most N_0 frontier nodes, let M_{j+1} be obtained from M_j by identifying any two frontier nodes with the same labels. A similar construction works if q_{j+1} is $E(q' \, U \, q)$.

If q_{j+1} is AFq, then let $t_1, \ldots, t_m$ be all the frontier nodes of M_j such that $AFq \in L(t_i)$. By induction, $m \leqslant N_0$. Let M_{j+1} be the result of replacing each t_i by $DAG[t_i, AFq]$ and again identifying any frontier nodes with the same labels. M_{j+1} satisfies the required properties: clearly, M_{j+1} has $\leqslant N_0$ frontier nodes. By Lemma 4.3, q_{j+1} is fulfilled for s in M_{j+1}. Finally, M_{j+1} has $\leqslant (j+1) N_0^2$ nodes since M_j has $\leqslant jN_0^2$ nodes (by induction) and each of the at most N_0 DAGs attached to M_j to form M_{j+1} has at most N_0 nodes. A similar construction works if q_{j+1} is $A(q' \, U \, q)$. ▮

The proof of Lemma 4.6 shows how to construct a finite pseudo-Hintikka structure for p_0 from $FRAG$s.

4.6. Lemma. *Let M be a pseudo-Hintikka structure for p_0 of size N_0. Then there is a Hintikka structure M'' for p_0 of size $\leqslant |p_0| \, N_0^3$.*

Proof. We first replace $L(s)$ by $L(s) \cap FL(p_0)$ for each state s in M. The resulting structure M' is still a pseudo-Hintikka structure for p_0 of size N_0. We construct M'' by splicing together $FRAG$s from M'. For each node s of M', $FRAG[s]$ will have at most one occurrence in M''. The construction is performed inductively, in stages. Let M_1 be $FRAG[s]$ for some state s of M' with $p_0 \in L(s)$. In general, to obtain M_{i+1} from M_i, do the following: For each frontier node s of M_i, if there is an interior node s' of M_i such that $L(s) = L(s')$ and $FRAG[s']$ is embedded in M_i then identify s and s'; otherwise, replace s by (a copy of) $FRAG[s]$ (as constructed in Lemma 4.5). The construction halts at $m =$ the least i such that the frontier of M_i is empty. Take $M'' = M_m$.

It is straightforward to check that M'' satisfies all the requrements for a Hintikka structure for p_0 with the possible exception of having unfulfilled eventualities (i.e., violations of (H17)–(H20). To see that this cannot happen, observe that, by the construction of M'',

(i) every node of M'' is contained in some fragment embedded in M'' which is of the form $FRAG[s]$ $(s \in M')$,

(ii) each frontier node of some fragment $FRAG[s]$ embedded in M'' is the root of still another fragment $FRAG[s']$ embedded in M''.

(To see this, recall that in the construction ot M'', a frontier node is identified with another node only if the other node is itself the root of some fragment $FRAG[s']$.

And if a frontier node is not identified with another node, it becomes the root of a fragment $FRAG[s']$ at the next stage of the construction.)

So suppose $AFq \in s$ for some node s of M''. By (i), s is contained in some fragment $FRAG[s']$. If AFq is fulfilled for s in $FRAG[s']$ we are done. Otherwise, by Lemma 4.3, $AFq \in t$ for every frontier node t of $FRAG[s']$ such that q does not occur somewhere on each path from s to t. By (ii), each such t is the root of $FRAG[t]$ embedded in M'' and AFq is thus fulfilled for t in M''. Using Lemma 4.3 we can easily show that AFq is also fulfilled for s in M''. The argument that eventualities of the form $A(p\ U\ q)$, $E(p\ U\ q)$, or EFq are fulfilled is similar and left to the reader.

To see that M'' is of the required size note that it consists of at most N_0 $FRAG$s, each containing at most $|p_0|\ N_0^2$ nodes. ∎

Returning to the proof of Theorem 4.1, we note that we can take $N_0 = 2^n$ to obtain the result. ∎

5. A Decision Procedure for Satisfiability in CTL

5.1. Theorem. *There is an algorithm for deciding whether a CTL formula is satisfiable which runs in deterministic time 2^{cn} for some constant $c > 0$.*

Proof. Given a formula p_0, we try to construct a pseudo-Hintikka structure for p_0 of size $\leqslant 2^{|p_0|}$. The algorithm is similar to Pratt's algorithm for deciding satisfiability of PDL formulae (cf. [17]). We proceed as follows:

(1) Define $s \subseteq FL(p_0)$ to be *maximal* if for every formula of the form $\neg q \in FL(p_0)$, either $\neg q \in s$ or $q \in s$. Let $S_0 = \{s \mid s \subseteq FL(p_0), s \text{ maximal}\}$. For each $s \in S_0$, define $L_0(s) = s$. Define a relation R_0 on $S_0 \times S_0$ such that for every $s, t \in S_0$, $(s, t) \in R_0$ iff

(a) $AXp \in s \Rightarrow p \in t$, and

(b) $\neg EXp \in s \Rightarrow \neg p \in t$.

(2) Let $S_1 = \{s \in S_0 \mid s \text{ satisfies (H1)–(H12)}\}$. Take L_1 and R_1 to be the restrictions of L_0 and R_0 to S_1 and $S_1 \times S_1$, respectively.

(3) Repeat for $i = 2, ..., N$, where N is the least number such that $|S_N| = |S_{N+1}|$: compute $M_i = (S_i, L_i, R_i)$ by taking $S_i = \{s \in S_{i-1} \mid s$ has an R_{i-1} successor, and if EXp (resp. $\neg AXp$, EFp, AFp, $E(p\ U\ q)$, $A(p\ U\ q)) \in s$, then (H13) (resp. (H1), (H17), (H18′), (H19), (H20′) holds for s in $M_{i-1}\}$, and L_i and R_i to be the restrictions of L_0 and R_0 to S_i and $S_i \times S_i$, respectively.

(4) Return "p_0 is satisfiable" iff for some $s \in S_N$, $p_0 \in s$.

Claim. The algorithm above is correct and can be implemented to run in time 2^{cn} for some constant $c > 0$, where $n = |p_0|$.

Proof. First note that $S_{i+1} \subseteq S_i$. Thus we must have $N \leqslant |S_0|$, and the algorithm terminates. Moreover, since $S_N = S_{N+1}$, no $s \in S_N$ can violate any of the conditions (H1)–(H17), (H18′), (H19), (H20′), and R_N must be a total relation on S_N. Thus, if $p_0 \in s$ for some $s \in S_N$, then M_N must be a pseudo-Hintikka structure for p_0, and hence p_0 is satisfiable by Theorem 4.1.

Conversely, if p_0 is satisfiable, say by M, let $M' = M/\equiv_{FL(p_0)} = (S', L', R')$. M' is a pseudo-Hintikka structure for p_0, and for all $s' \in S'$, $L'(s')$ is maximal. Let $f\colon S' \to S_0$ via $f(s') = L'(s')$. Then it is easily checked that $(s, t) \in R' \Rightarrow (f(s), f(t)) \in R_0$. We can then show by induction on i that for all $i \leqslant N$, we have $f(S') \subseteq S_i$, and $(s, t) \in R' \Rightarrow (f(s), f(t)) \in R_i$. (This can be argued by a case-by-case analysis on how nodes of S_i are eliminated. For instance, if $AFp \in s$, where $s \in S_i$ and for some $s' \in S'$, $s = f(s')$, then s cannot be eliminated due to a violation of (H18′) because the image under f of the appropriate fragment contained in M' is a fragment contained in M_i. The details are straightforward and left to the reader.) It follows that for some $s \in S_N$, we will have $p_0 \in s$.

Now we consider implementation details. Since $|FL(p_0)| \leqslant 2|p_0|$ $(=2n)$, S_0 has $\leqslant 2^{2n}$ members. Step (1) can clearly be done in time quadratic in the size of S_0, while step (2) can be done in linear time. Step (3) will be repeated at most $|S_0|$ times. Thus, it suffices to establish that each check in step (3) can be done in time polynomial in the number of nodes remaining in the graph. The case of EXp or $\neg AXp$ is straightforward. We sketch the algorithm for $A(p\ U\ q)$

(1) Mark all nodes s for which $q, A(p\ U\ q) \in s$.

(2) Mark all unmarked nodes s with $p, A(p\ U\ q) \in s$ such that for each $p' \in s$ of the form EXq' or $\neg AXq'$ there is a marked R-successor s' of s with $q' \in s'$ or $\neg q' \in s'$, respectively. Repeat this step until no more nodes can be marked.

(3) Eliminate all unmarked nodes t such that $A(p\ U\ q) \in t$.

We leave it to the reader to check that this algorithm is correct and has the desired complexity. Similar algorithms work for AFq, EFq, and $E(p\ U\ q)$. ∎

5.2. *Remark.* The proof that deterministic exponential time is a lower bound for PDL ([9]) carries over directly to UB^- (and hence both UB and CTL). Thus the decision procedure given above is essentially the best we can get.

6. A Complete Axiomatization for CTL

6.1. Consider the following axioms and rules of inference:

Axioms

(Ax1) All (substitution instances of) tautologies of propositional logic
(Ax2) $EFp \equiv E(true\ U\ p)$
(Ax3) $AFp \equiv A(true\ U\ p)$

(Ax4) $EX(p \vee q) \equiv EXp \vee EXq$
(Ax5) $AXp \equiv \neg EX \neg p$
(Ax6) $E(p\,U\,q) \equiv q \vee (p \wedge EXE(p\,U\,q))$
(Ax7) $A(p\,U\,q) \equiv q \vee (p \wedge AXA(p\,U\,q))$
(Ax8) $EXtrue \wedge AXtrue$.

Rules of Inference.

(R1) $p \Rightarrow q \vdash EXp \Rightarrow EXq$
(R2) $r \Rightarrow (\neg q \wedge EXr) \vdash r \Rightarrow \neg A(p\,U\,q)$
(R3) $r \Rightarrow [\neg q \wedge AX(r \vee \neg E(p\,U\,q))] \vdash r \Rightarrow \neg E(p\,U\,q)$
(R4) $p, (p \Rightarrow q) \vdash q$ (modus ponens).

These axioms and rules of inference are clearly sound and are also complete as shown below. If we replace p by *true* in (Ax6), (Ax7), (R2), and (R3) above and use the equivalences in (Ax2) and (Ax3) we get a complete axiomatization of UB equivalent to the one given in [4]. (For example, (Ax7) becomes $AFq \equiv q \vee AXAFq$.)

6.2. THEOREM. *The above set of axioms and rules of inference is complete for CTL.*

Proof. We say that a formula p is *provable*, and write $\vdash p$, if there exists a finite sequence of formulae, ending with p, such that each formula is an instance of an axiom scheme or follows from previous formulas by one of the inference rules. A formula p is *consistent* if not $\vdash \neg p$, i.e., if $\neg p$ is not provable. We want to show that any valid CTL formula is provable. It suffices to show that any consistent formula is satisfiable.

So suppose p_0 is a consistent CTL formula. We try to construct a model for p_0 just as in the proof of Theorem 5.1. For each $s \in S_0$, define the formula p_s as the conjunction of the formulae in s; i.e., $p_s = \bigwedge_{q \in s} q$. Note that since s is maximal, if $q \in FL(p_0)$ then $q \in s$ iff $\vdash p_s \Rightarrow q$.

We will show that if a state $s \in S_0$ is eliminated in the algorithm in the proof of Theorem 5.1, then p_s is inconsistent. Once we have shown this, we can argue as follows: It is easy to check by propositional reasoning that for any $q \in FL(p_0)$ we have

$$\vdash q \equiv \bigvee_{\{s \mid q \in s, p_s \text{ consistent}\}} p_s \quad \text{and} \quad \vdash true \equiv \bigvee_{\{s \mid s \in S_0, p_s \text{ consistent}\}} p_s. \qquad (*)$$

In particular. $\vdash p_0 \equiv \bigvee_{\{s \mid p_0 \in s, p_s \text{ consistent}\}} p_s$, so if p_0 is consistent, some p_s is consistent. This particular s will not be eliminated in the course of our construction. Thus, at the end we will be left with a pseudo-Hintikka structure for p_0, so by Theorem 4.1, p_0 is satisfiable.

We now show, by induction on when a state is eliminated, that if state s is eliminated then $\vdash \neg p_s$:

(1) It is easy to check that if s is eliminated in step 2, then p_s must be incon-

sistent due to (Ax1)-(Ax7) and the fact that for each $q \in FL(p_0)$, either $\vdash p_s \Rightarrow q$ or $\vdash p_s \Rightarrow \neg q$.

(2) *Claim.* If $(s, t) \notin R_0$ as constructed in step 1, $p_s \wedge EXp_t$ is inconsistent.

Proof. If $AXp \in s$ and $p \notin t$, then $\vdash p_s \Rightarrow AXp$ and $\vdash p_t \Rightarrow \neg p$. By (R1), $\vdash EXp_t \Rightarrow EX\neg p$. Thus $\vdash (p_s \wedge EXp_t) \Rightarrow AXp \wedge EX\neg p$. But by (Ax5) it follows that $AXp \wedge EX\neg p$, and hence $p_s \wedge EXp_t$, is inconsistent. The proof in the other case is similar.

(3) *Claim.* If p_s is consistent, then s is not eliminated at step 3.

Proof.
$$\begin{aligned} \vdash p_s &\equiv p_s \wedge EXtrue && \text{by (Ax8)} \\ &\equiv p_s \wedge EX(\textstyle\bigvee_{p_t \text{ consistent}} p_t) && \text{by } (*), \text{(R1)} \\ &\equiv p_s \wedge (\textstyle\bigvee_{p_t \text{ consistent}} EXp_t) && \text{by (Ax4)} \\ &\equiv \textstyle\bigvee_{p_t \text{ consistent}} (p_s \wedge EXp_t) && \text{by (Ax1).} \end{aligned}$$

Thus, if p_s is consistent, $p_s \wedge EXp_t$ must be consistent for some t with p_t consistent. By (2) above, $(s, t) \in R_0$. By the induction hypothesis, t is not eliminated and s will have an R_0-successor. Thus p_s will not be eliminated by step 3.

(4) *Claim.* If p_s is consistent, then s is not eliminated at step 4.

Proof. (a) If $EXp \in s$, then by the same reasoning used above $\vdash p_s \equiv \bigvee_{\{t \mid p_t \text{ consistent}, p \in t\}} (p_s \wedge EXp_t)$. Thus, for some t with p_t consistent and $p \in t$, we have $(s, t) \in R_0$, so s satisfies (H13).

(b) A similar proof shows that if $\neg AXp \in s$, s satisfies (H16).

(c) Suppose s is eliminated at step (4) on account of (H19) failing at s with respect to $E(p\,U\,q)$. We will show that p_s is inconsistent. Let $V = \{t \mid E(p\,U\,q) \in t$ and t is eliminated at step (4) because (H19) fails for $E(p\,U\,q)\}$. By assumption, $s \in V$.

Since (H19) fails, $\vdash p_t \Rightarrow \neg q$ for each $t \in V$. Let $r = \bigvee_{t \in V} p_t$. Note we also have $\vdash r \Rightarrow \neg q$.

Suppose we can show $\vdash r \Rightarrow AX(r \vee \neg E(p\,U\,q))$. Then $\vdash r \Rightarrow \neg q \wedge AX(r \vee \neg E(p\,U\,q))$. By (R3), $\vdash r \Rightarrow \neg E(p\,U\,q)$. Since $s \in V$, $\vdash p_s \Rightarrow r$, so by (R4), $\vdash p_s \Rightarrow \neg E(p\,U\,q)$. By assumption we have $E(p\,U\,q) \in s$, so p_s must be inconsistent, as desired.

In order to show $\vdash r \Rightarrow AX(r \vee \neg E(p\,U\,q))$, it suffices to show that for each $t \in V$, $\vdash p_t \Rightarrow AX(r \vee \neg E(p\,U\,q))$. Suppose not. Then for some $t \in V$, $p_t \wedge EX(\neg r \wedge E(p\,U\,q))$ is consistent. But $\neg r \equiv \bigvee_{t' \notin V} p_{t'}$, so by (Ax4), $p_t \wedge EX(p_{t'} \wedge E(p\,U\,q))$ is consistent for some $t' \notin V$. It follows that both $p_t \wedge EXp_{t'}$ and $p_{t'} \wedge E(p\,U\,q)$ are consistent. The former implies $(t, t') \in R_0$ by (2) above, while the latter implies $E(p\,U\,q) \in t'$ since one of $E(p\,U\,q)$, $\neg E(p\,U\,q) \in t$ by maximality. But if $E(p\,U\,q) \in t'$ and $t' \notin V$, then (H19) must hold for t'. Since $(t, t') \in R_0$, (H19) must also hold for t, contradicting the fact that $t \in V$.

A similar argument shows that if $EFp \in s$ and s is eliminated at step (4) because (H17) is not satisfied, p_s is inconsistent.

(d) Suppose s is eliminated at step (4) on account of (H20′) failing at s with respect to $A(p\,U\,q)$. Again we show that p_s is inconsistent.

Let $W = \{t \mid t$ is eliminated at step (4) because (H20′) fails for $A(p\,U\,q)\}$. By assumption, $s \in W$. Note that by (Ax7), $\vdash p_t \Rightarrow AXA(p\,U\,q) \wedge \neg q$ for each $t \in W$. Let $r = \bigvee_{t \in W} p_t$. Clearly $\vdash r \Rightarrow \neg q$.

Suppose we can show $\vdash r \Rightarrow EXr$. Then, $\vdash r \Rightarrow \neg q \wedge EXr$. By (R2), $\vdash r \Rightarrow \neg A(p\,U\,q)$. Since $s \in W$, $\vdash p_s \Rightarrow r$ and thus $\vdash p_s \Rightarrow \neg A(p\,U\,q)$. It follows that p_s is inconsistent.

In order to show $\vdash r \Rightarrow EXr$, it suffices to show that for each $t \in W$, $\vdash p_t \Rightarrow EXr$. Given $t \in W$, let $E_t = \{q \mid EXq \in t\} \cup \{\neg q \mid \neg AXq \in t\} \cup \{true\}$, and let $A_t = \{q \mid AXq \in t\} \cup \{\neg q \mid \neg EXq \in t\}$.

For each $q' \in E_t$, define $f_{q'} = q' \wedge (\bigwedge_{q'' \in A_t} q'')$ and let $X_{q'} = \{t' \mid (t, t') \in R_0, \vdash p_{t'} \Rightarrow q'\}$. It is easy to check that

(i) $\vdash p_t \Rightarrow EXf_{q'}$ and

(ii) $\vdash f_{q'} \equiv \bigvee_{t' \in X_{q'}} p_{t'}$.

Note also that for each $t' \in X_{q'}$, $A(p\,U\,q) \in t'$ (since $AXA(p\,U\,q) \in t$). Now, if for each $q' \in E_t$ there is a $t' \in X_{q'}$ such that $A(p\,U\,q)$ satisfies (H20′) at t', then we see that $A(p\,U\,q)$ satisfies (H20′) at t as well, which contradicts the assumption that $t \in W$. So it must be that for some $q' \in E_t$ and for all $t' \in X_{q'}$, we have $t' \in W$. For this q', $X_{q'} \subseteq W$. By (ii) above, it follows that $\vdash f_{q'} \Rightarrow r$. Using (i), we obtain $\vdash p_t \Rightarrow EXr$.

A similar argument applies if $AFq \in s$. We have now shown that only states s with p_s inconsistent are eliminated, thus completing our proof. ∎

6.3. *Remark.* As we mentioned in 2.4, the condition that R be total can be removed from our definitions of model, Hintikka structure, and pseudo-Hintikka structure. But in this case, Lemma 2.6.(3) must be modified to read $\models AFp \equiv p \vee (AXAFp \wedge EX\,true)$. The clause $EXtrue$ must also be added in 2.6.(5), (H7), (H11), and (Ax7) and removed from (Ax8). We can then eliminate step (3) in both Theorems 5.1 and 6.2. All other results go through unchanged.

7. Tableau Techniques

7.1. *Constructing the Tableau*

The algorithm for deciding satisfiability presented in Theorem 5.1 has a worst-case running time of 2^{cn}. This is the best we can do in light of the remarks in 5.2. However, its average-case performance is also 2^{cn} since the first step involves creating all the subsets of $FL(p_0)$. Just as for DPDL (cf. [3]) there is a "bottom-up" procedure for constructing a pseudo-Hintikka structure which is likely to perform better in practice.

We say that an *elementary formula* is one of the form P, $\neg P$, EXp, AXp, $\neg EXp$, or $\neg AXp$. We classify each *nonelementary formula* as either a conjunctive formula $\alpha \equiv \alpha_1 \wedge \alpha_2$ or a disjunctive formulae $\beta \equiv \beta_1 \vee \beta_2$. Clearly, $p \wedge q$ is an α-formula and $\neg(p \wedge q)$ (which is equivalent to $\neg p \vee \neg q$) is a β-formula. A temporal operator is classified as either α or β based on its fixpoint characterization (cf. [6]) as in Lemma 2.6. Thus, $AFp \equiv p \vee AXAFp$ is a β-formula and $\neg AFp \equiv \neg p \wedge \neg AXAFp$ is an α-formula. The table below summarizes the classification of nonelementary formulae as either α or β:

α - $p \wedge q$	α_1 - p	α_2 - q
α - $\neg\neg p$	α_1 - p	α_2 - p
α - $\neg EFp$	α_1 - $\neg p$	α_2 - $\neg EXEFp$
α - $\neg AFp$	α_1 - $\neg p$	α_2 - $\neg AXAFp$
β - $\neg(p \wedge q)$	β_1 - $\neg p$	β_2 - $\neg q$
β - EFp	β_1 - p	β_2 - $EXEFp$
β - AFp	β_1 - p	β_2 - $AXAFp$
β - $E(p\,U\,q)$	β_1 - p	β_2 - p, $EXE(p\,U\,q)$
β - $A(p\,U\,q)$	β_1 - q	β_2 - p, $AXA(p\,U\,q)$
β - $\neg E(p\,U\,q)$	β_1 - $\neg q$, $\neg p$	β_2 - $\neg q$, $\neg EXE(p\,U\,q)$
β - $\neg A(p\,U\,q)$	β_1 - $\neg q$, $\neg p$	β_2 - $\neg q$, $\neg AXA(p\,U\,q)$

Given a formula p_0, we proceed to build a structure in stages:

(1) Label the "root" node by $\{p_0\}$.

(2) Inductively assume we have constructed a graph with nodes labelled by subsets of $FL(p_0)$. At each node certain formulae in the label are marked "expanded." For every frontier node labelled by $\Gamma \subseteq FL(p_0)$, choose some nonelementary formula q which is not marked, and expand it according to the table above: if q is an α-formula, create one son of this node labelled by $\Gamma \cup \{\alpha_1, \alpha_2\}$ and mark q. If q is a β-formula, create two sons of this node, one labelled $\Gamma \cup \{\beta_1\}$ and the other $\Gamma \cup \{\beta_2\}$. In the label of each son, mark q. As usual, any two nodes with the same label and the same formulae marked expanded are identified. Thus, there are at most 2^{4n} nodes, where $n = |p_0|$.

(3) If all the nonelementary formulae at a node are marked, this node is called a *state*. Let $EXq_1,..., EXq_k$, $\neg AXq_{k+1},..., \neg AXq_m$, $AXr_1,..., AXr_i,..., \neg EXr_{i+1},..., \neg EXr_n$ be the nonatomic elementary formulae in the label of a state s. Create $m+1$ sons of s, labelled by

$$\begin{array}{ll} \{r_1,..., r_i, \neg r_{i+1},..., \neg r_n, q_j\}, & j = 1,..., k, \\ \{r_1,..., r_i, \neg r_{i+1},..., \neg r_n, \neg q_j\}, & j = k+1,..., m, \\ \{r_1,..., r_i, \neg r_{i+1},..., \neg r_n\}, & \text{respectively.} \end{array}$$

(4) Repeat steps (2) and (3) until no more nodes can be added.

From this structure we create a *tableau*, $M' = (S_0, L_0, R_0)$. S_0 consists of exactly those nodes which were states in the construction above. For $s \in S_0$, $L_0(s)$ is the label on s. For $s, t \in S_0$, $(s, t) \in R_0$ iff there is a path from s to t in the above graph which does not go through any other states. For a node labelled by Γ, define $p_s = \bigwedge_{r \in \Gamma} r$.

Once we have constructed the tableau, we just repeat steps (1), (4), and (5) of the algorithm in the proof of Theorem 5.1; if there is a state containing p_0 which is not eliminated, we have constructed a pseudo-Hintikka structure for p_0, so p_0 is satisfiable.

For the converse, we can show as in Theorem 6.2 that if a state is eliminated then $\models \neg p_s$. (Note that this will also give us another proof of the completeness of the axiomatization presented in 6.1.) Not surprisingly, the details of this proof are similar to those in 6.2; we omit them here. ∎

8. Expressiveness

8.1. Theorem. *AFp is not expressible in* UB^-.

Proof. An argument completely analogous to that given in [9] for PDL shows that the quotient construction preserves satisfiability for UB^- formulae. In the notation of Section 3.5, if p is a UB^- formula and $H \supseteq FL(p)$, then for all $q \in FL(p)$ and all models M, we have $M, s \models q$ iff $M/\equiv_H, [s] \models q$. (That the proof should be analogous to PDL should not be too surprising. We can view CTL models as models of PDL with one primitive program r, whose semantics are given by the relation R. If we now translate CTL formulas into PDL formulas via the translation $EXp \rightarrow \langle r \rangle p$ and $EFp \rightarrow \langle r^* \rangle p$, then a UB^- formula q is satisfiable in a CTL model M iff the translated formula is satisfiable at the same state of M when M is viewed as a PDL model.) Now suppose AFP is equivalent to the UB^- formula p. Let M be the model from the proof of Theorem 3.6, and let $H = FL(p) \cup \{P\}$. Since $M, s \models AFP$ for all states s in M, and p is equivalent to AFP by hypothesis, we must have $M, s \models p$ for all states s in M. Since the quotient construction preserves satisfiability for UB^- formulae, we must also have $M/\equiv_H, [s] \models p$ for all s. But the proof of Theorem 3.6 shows that $M/\equiv_H, [s_i] \models \neg AFP$, contradicting the equivalence of p and AFP. ∎

8.2. Theorem. *$E(FP \wedge GQ)$ is not expressible in UB.*

Proof. We will define inductively two sequences of models $M_1, M_2, M_3, \ldots$, and $N_1, N_2, N_3, \ldots$, such that for all i, we have $M_i, s_i \models E(FP \wedge GQ)$ and $N_i, s_i \models \neg E(FP \wedge GQ)$. We will show that UB is unable to distinguish between the two sequences of models, i.e., for all UB formulae p with $|p| \leqslant i$, $M_i, s_i \models p$ iff $N_i, s_i \models p$. To see that the result follows suppose that $E(FP \wedge GQ)$ is equivalent to some UB formula p. Then $M_{|p|}, s_{|p|} \models p$ iff $N_{|p|}, s_{|p|} \models p$ contradicting the fact that $M_{|p|}, s_{|p|} \models E(FP \wedge GQ)$ and $N_{|p|}, s_{|p|} \models \neg E(FP \wedge GQ)$. The details of the proof are given below.

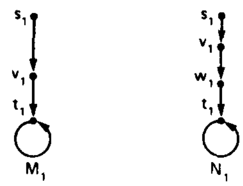

FIGURE 4

Define M_1, N_1 to have the graphs shown below in Fig. 4. where $s_1 \models \neg P \wedge Q$, $v_1 \models P \wedge Q$, $t_1 \models \neg P \wedge Q$, and $w_1 \models P \wedge \neg Q$.

Suppose we have defined M_i and N_i. Then M_{i+1} has the graph shown in Fig. 5, where $s_{i+1} \models \neg P \wedge Q$, $t_{i+1} \models \neg P \wedge Q$, $u_{i+1} \models P \wedge \neg Q$, M_i', M_i'' are copies of M_i, and N_i'' is a copy of N_i. N_{i+1} is defined similarly, except that M_i' is replaced by N_i', a copy of N_i. It is straightforward to show by induction on i that

(1) $M_i, s_i \models E(FP \wedge GQ)$ (since the path through M_{i-1}' satisfies $FP \wedge GQ$) and $N_i, s_i \models \neg E[FP \wedge GQ]$ (since the state u_i prevents the possibility of a path satisfying $FP \wedge GQ$ going through M_i'' or N_i').

(2) Each path in M_i and in N_i ends in a self-loop through some state t_j.

We now show by induction on $|p|$, that for all UB formulae p, if $|p| \leqslant i$ then

$$M_i, s_i \models p \text{ iff } N_i, s_i \models p. \qquad (**)$$

Since $\models AXq \equiv \neg EX \neg q$ and $\models AFq \equiv \neg EG \neg q$, we can take EX, EG, and EF as the primitive temporal operators in our induction. The cases where p is an atomic formula, a conjunction $p_1 \wedge p_2$, or a negation $\neg p_1$ are easy and left to the reader. If p is of the form EXq,

$M_{i+1}, s_{i+1} \models EXq$

iff $M_{i+1}, s \models q$, where s is t_{i+1}, u_{i+1}, or s_i',

iff $N_{i+1}, s \models q$, where s is t_{i+1}, u_{i+1}, or s_i', (see below for details)

iff $N_{i+1}, s_{i+1} \models EXq$.

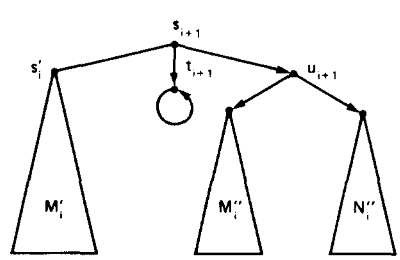

FIGURE 5

The first two cases of the third equivalence are obvious from the definitions of M_{i+1}, N_{i+1}; when $s = s_i'$, we have

$$\begin{array}{ll} & M_{i+1}, s_i' \models q \\ \text{iff} & M_i, s_i \models q \\ \text{iff} & N_i, s_i \models q \quad \text{(by the induction assumption)} \\ \text{iff} & N_{i+1}, s_i' \models q. \end{array}$$

If p is of the form EGq,

$$\begin{array}{ll} & M_{i+1}, s_{i+1} \models EGq \\ \text{iff} & \text{there is a path in } M_{i+1} \text{ starting at } s_{i+1} \text{ all of whose nodes satisfy } q \\ \text{iff} & M_{i+1}, s_{i+1} \models q \quad \text{and} \quad M_{i+1}, t_{i+1} \models q \text{ (by (2) above)} \\ \text{iff} & N_{i+1}, s_{i+1} \models q \quad \text{and} \quad N_{i+1}, t_{i+1} \models q \text{ (by the induction hypothesis)} \\ \text{iff} & \text{there is a path in } N_{i+1} \text{ starting at } s_{i+1} \text{ all of whose nodes satisfy } q \\ \text{iff} & N_{i+1}, s_{i+1} \models EGq. \end{array}$$

Finally, suppose p is of the form EFq and $M_{i+1}, s_{i+1} \models EFq$. Then $M_{i+1}, s \models q$ for some state s in M_{i+1}. There are several cases to consider:

(a) If $s = s_{i+1}$, then $N_{i+1}, s_{i+1} \models q$ by the induction assumption.

(b) Of $s = t_{i+1}$ or u_{i+1}, then $N_{i+1}, s \models q$ since the world "below" t_{i+1} or u_{i+1} looks the same in M_{i+1} and N_{i+1}.

(c) If s is in M_i'' or N_i'', then s is also a state in N_{i+1}, so $N_{i+1}, s \models q$.

(d) If s is some s' in M_i' then there is a corresponding s'' in M_i'', and thus in N_{i+1}, with $N_{i+1}, s'' \models q$.

In each case we conclude that $N_{i+1}, s_{i+1} \models EFq$. The converse is identical (upon interchanging the roles of M and N).

This comples the proof of (**) and the theorem follows. ∎

The following result for branching time is analogous to the corresponding result for linear time due to Kamp (cf. [11]).

8.3. THEOREM. *$E(p\ U\ q)$ is not expressible in UB^+.*

Proof. We define two (doubly-indexed) sequences of models M_{ki} and N_{ki}. The graph of each model is just a straight line:

$$\underset{s_0}{\bullet} \longrightarrow \underset{s_1}{\bullet} \longrightarrow \underset{s_2}{\bullet} \longrightarrow \underset{s_3}{\bullet} \cdots$$

For M_{ki} we have

$$s_j \models P \wedge \neg Q \quad \text{if } j \neq i + tk \text{ for some } t,$$
$$s_j \models P \wedge Q \quad \text{if } j = i + tk \text{ for some even } t,$$
$$s_j \models \neg P \wedge \neg Q \quad \text{if } j = i + tk \text{ for some odd } t.$$

For N_{ki},

$$s_j \models P \wedge \neg Q \quad \text{if } j \neq i + tk \text{ for some } t,$$
$$s_j \models \neg P \wedge \neg Q \quad \text{if } j = i + tk \text{ for some even } t,$$
$$s_j \models P \wedge Q \quad \text{if } j = i + tk \text{ for some odd } t.$$

Thus, for both M_{ki} and N_{ki}, $P \wedge \neg Q$ is true at all states not congruent to $i \bmod k$. $P \wedge Q$ and $\neg P \wedge \neg Q$ alternate being true at states congruent to $i \bmod k$. Since $P \wedge Q$ is true first in M_{ki}, while $\neg P \wedge \neg Q$ is true first in N_{ki}, we have $M_{ki}, s_0 \models E(P\, U\, Q)$. However, by a straightforward induction, we can show for all UB formulae p that if $|p| \leqslant i$, $M_{ki}, s_0 \models p$ iff $N_{ki}, s_0 \models p$. The proof uses the observation that $M_{ki}, s_0 \models p$ iff $N_{ki}, s_k \models p$. We omit the details here. Now we can argue just as in the previous theorem that $E(P\, U\, Q)$ is not equivalent to any UB formula. Moreover, since UB^+ reduces to UB in M_{ki} and N_{ki} (since there is only one path, we have $E(p \wedge q) \equiv Ep \wedge Eq$, etc.) $E(P\, U\, Q)$ is not equivalent to any UB^+ formula. ∎

Although Theorem 8.2 shows that UB is less expressive than UB^+, this result does not extend to CTL as the following theorem shows (cf. [6, Theorem 5.1]).

8.4. THEOREM. *For all CTL^+ formulae p, there is a CTL formula p' such that $\models p \equiv p'$. Moreover, $|p'| \leqslant 2^{|p|\log|p|}$.*

Proof. We describe an algorithm for translating a CTL^+ formula p_0 into a equivalent CTL formula p_0'. We can assume without loss of generality that A and F do not occur in p_0 since $Aq \equiv \neg E \neg q$ and $Fq \equiv \mathit{true}\; U\, q$. We can then reduce the problem to one of translating a CTL^+ formula with at most one E by recursively applying the algorithm to nested subformulae containing an E. So we can assume p_0 is of the form Eq_0, where q_0, a path formula, is a Boolean combination of subformulae of the form $p\, U\, q$, $\neg(p\, U\, q)$, Xr, and $\neg Xr$, where p, q, and r are CTL formulae (found by recursive applications of the algorithm).

Observe that the following equivalences hold:

(1) $\models \neg Xr \equiv X \neg r$

(2) $\models \neg(p\, U\, q) \equiv [(p \wedge \neg q)\, U\, (\neg p \wedge \neg q)] \vee G \neg q$

(3) $\models E(p \vee q) \equiv Ep \vee Eq$

(4) $\models X(p \wedge q) \equiv Xp \vee Xq$

(5) $\models G(p \wedge q) \equiv Gp \wedge Gq$

(6) $\models E(\wedge_{j=1,\ldots,n}(p_j \, U \, q_j) \wedge Xr_1 \wedge Gr_2)$
$\equiv \bigvee_{J \subseteq \{1,\ldots,n\}} [\bigwedge_{j \in J} q_j \wedge r_2 \wedge EX(r_1 \wedge E(\bigwedge_{j \notin J} p_j \, U \, q_j \wedge Gr_2)]$

(7) $\models E(\wedge_{j=1,\ldots,n}(p_j \, U \, q_j) \wedge Gr)$
$\equiv \bigvee \{E((\bigwedge_j p_j \wedge \mathrm{r}) \, U \, (q_{\pi(1)} \wedge E((\bigwedge_{j \neq \pi(1)} p_j \wedge r)$
$U \, (q_{\pi(2)} \wedge E((\bigwedge_{j \neq \pi(1), \pi(2)} p_j \wedge r) \, U \, (q_{\pi(3)} \wedge \cdots$
$\wedge E((p_{\pi(n)} \wedge r) \, U \, (q_{\pi(n)} \wedge EGr))))))))$: π is a permutation of $\{1,\ldots,n\}\}$.

Intuitively, the right side of the last equivalence is a disjunction over all the possible orders in which the q_i's can be satisfied along the path.

We proceed as follows: Using DeMorgan's laws, drive negations inward until q_0 is composed of conjunctions and disjunctions of formulae of the form $p \, U \, q$, $\neg(p \, U \, q)$, Xr, and $\neg Xr$. After applying equivalences (1) and (2), we assume q_0 is made up of disjunctions and conjunctions of formulae of the form $p \, U \, q$, Xr_1, and Gr_2. We put this into disjunctive normal form and apply equivalences (3), (4), and (5). We have now reduced the problem to one of translating a formula Eq', where q' is of the form

$$\bigwedge_{j=1,\ldots,n} (p_j \, U \, q_j) \wedge Xr_1 \wedge Gr_2.$$

Using equivalence (6), we can eliminate the Xr term from consideration. Finally, using equivalence (7) gives us a formula in CTL.

Note that using equivalence (7) introduces a factorial blowup in the sign of the formula. We can show that this is the worst blowup that happens in the translation process. Since $n! = O(2^{n \log n})$, we can show that $|p'| \leqslant 2^{|p| \log |p|}$. We omit details here. ∎

Thus we get the following hierarchy of branching time logics (where $<$ indicates "strictly less expressive than" and $\equiv$ indicates "exactly as expressive as"):

$$\mathrm{UB}^- < \mathrm{UB} < \mathrm{UB}^+ < \mathrm{CTL} \equiv \mathrm{CTL}^+.$$

Finally, putting together Theorems 8.4 and 5.1 we get

8.5. THEOREM. *There is a decision procedure for satisfiability of CTL^+ formulas which runs in time $2^{2^{cn \log(n)}}$ for some $c > 0$.*

9. CONCLUSION

We have shown that, while the Fischer–Ladner quotient construction fails to preserve satisfiability of CTL (and UB) formulae, it still provides enough useful information to give a decision procedure for satisfiability of CTL formulae that runs in single exponential time. CTL is sufficiently expressive to allow the specification of many interesting synchronization problems. A method of automatically synthesizing solutions to these problems based on a variant of the tableau-based decision procedure of Section 7 is described in [5]. We also classify the relative expressive power of a number of languages obtained by extending or restricting the CTL syntax.

These issues are further studied in [7]. There we define a language CTL* which contains CTL^+ as a proper sublanguage, and obtain a number of results similar in spirit to those of Section 8. We also show that CTL* is closely related to the logic. MPL of Abrahamson [1]. MPL is shown in [1] to have a double exponential time decision procedure, and it would be nice if we could apply these techniques to CTL^+ and CTL* as well. However, the semantics of MPL differs in one crucial way from those of the languages we have been studying: the computation paths are not necessarily generated by a binary relation. Thus they do not necessarily have the "limit closure" property; i.e., if all the prefixes of a path are in the structure, the path itself is present in the structure (cf. [8]). Thus there seems no obvious way of transferring results on MPL to CTL^+ or CTL*. We refer the reader to [7] for a more detailed discussion of these points.

Acknowledgment

We would like to thank Kata Carbone for the fine job of typing the manuscript.

References

1. K. Abrahamson, "Decidability and Expressiveness of Logics of Processes," PhD thesis, Univ. of Washington, 1980.
2. M. Ben-Ari, personal communication.
3. M. Ben-Ari, J. Y. Halpern, and A. Pnueli, Finite models of deterministic propositional dynamic logic, *in* "Int. Colloq. Automata Lang. and Programming, 1981," pp. 249–263; revised version: Deterministic propositional dynamic logic: Finite models, complexity, and completeness, *J. Comput. System Sci.* **25**, No. 3 (1982), 402–417.
4. M. Ben-Ari, Z. Manna, and A. Pnueli, The temporal logic of branching time, *in* "Proc. Ann. ACM Sympos. Principles of Programming Languages, 1981," pp. 164–176.
5. E. A. Emerson and E. M. Clarke, Using branching time logic to synthesize synchronization skeletons, *Sci. Comput. Programming* **2** (1982), 241–266.
6. E. A. Emerson and E. M. Clarke, Characterizing correctness properties of parallel programs as fixpoints, *in* "Int. Colloq. Automata Lang. and Programming, 1980," pp. 169–181.
7. E. A. Emerson and J. Y. Halpern, "Sometimes" and "Not Never" revisited: On branching versus linear time, *in* "Proc. Ann. ACM Sympos. Principles of Programming Languages, 1983," pp. 127–140.
8. E. A. Emerson, "Alternative Semantics for Temporal Logics," Tech. Report TR-182, Univ. of Texas, 1981; *Theor. Comput. Science* **26**, Nos. 1, 2 (1983), 121–130.
9. M. J. Fischer and R. E. Ladner, Propositional dynamic logic of regular programs, *J. Comput. System Sci.* **18**, No. 2 (1979), 194–211.

10. D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi, On the temporal analysis of fairness, *in* "Proc. Annual ACM Sympos. Principles of Programming Languages, 1980," pp. 163–173.
11. H. W. Kamp, "Tense Logic and the Theory of Linear Order," PhD thesis, UCLA, 1968.
12. D. Kozen and R. Parikh, An elementary proof of the completeness of PDL, *Theor. Comput. Science* **14**, No. 1 (1981), 113–118.
13. L. Lamport, "Sometimes" is sometimes "not never," *in* "Proc. Annual ACM Sympos. Principles of Programming Languages, 1980," pp. 174–183.
14. Z. Manna and A. Pnueli, The modal logic of programs, *in* "Int. Colloq. Automata Lang. and Programming, 1979," pp. 385–410.
15. A. Pnueli, The temporal logic of programs, *in* "Proc. IEEE Annual Sympos. Forind. Comput. Sci, 1977," pp. 46–57.
16. V. R. Pratt, "Applications of Modal Logic to Programming, MIT LCS TM-116, 1978.
17. V. R. Pratt, Models of program logics, *in* "Proc. IEEE Annual Sympos. Found. Comput. Sci., 1979," pp. 15–122.
18. R. M. Smullyan, "First-Order Logic," Springer-Verlag, Berlin, 1967.
19. R. S. Streett, "Propositional Dynamic Logic of Looping and Converse," PhD thesis, MIT, 1981.